Regression Predict Student Solution

Honour Code

I {**MASILO, RAMATSEBA**}, confirm - by submitting this document - that the solutions in this notebook are a result of my own work and that I abide by the EDSA honour code.

Non-compliance with the honour code constitutes a material breach of contract.

Predict Overview: Spain Electricity Shortfall Challenge

The government of Spain is considering an expansion of it's renewable energy resource infrastructure investments. As such, they require information on the trends and patterns of the countries renewable sources and fossil fuel energy generation. Your company has been awarded the contract to:

- 1. analyse the supplied data;
- 2. identify potential errors in the data and clean the existing data set;
- 3. determine if additional features can be added to enrich the data set;
- 4. build a model that is capable of forecasting the three hourly demand shortfalls;
- 5. evaluate the accuracy of the best machine learning model;
- 6. determine what features were most important in the model's prediction decision, and
- 7. explain the inner working of the model to a non-technical audience.

Formally the problem statement was given to you, the senior data scientist, by your manager via email reads as follow:

> In this project you are tasked to model the shortfall between the energy generated by means of fossil fuels and various renewable sources - for the country of Spain. The daily shortfall, which will be referred to as the target variable, will be modelled as a function of various city-specific weather features such as `pressure`, `wind speed`, `humidity`, etc. As with all data science projects, the provided features are rarely adequate predictors of the target variable. As such, you are required to perform feature engineering to ensure that you will be able to accurately model Spain's three hourly shortfalls.

On top of this, she has provided you with a starter notebook containing vague explanations of what the main outcomes are.

Table of Contents

1. Importing Packages

Back to Table of Contents

> ⚡ **Description: Importing Packages** ⚡
>
> In this section you are required to import, and briefly discuss, the libraries that will be used throughout your analysis and modelling.

In [1]:
```
pip install missingno
```

```
Requirement already satisfied: missingno in c:\users\mohal\anaconda3\lib\site-packages (0.5.2)
Requirement already satisfied: numpy in c:\users\mohal\anaconda3\lib\site-packages (from missingno) (1.26.4)
Requirement already satisfied: matplotlib in c:\users\mohal\anaconda3\lib\site-packages (from missingno) (3.8.0)
Requirement already satisfied: scipy in c:\users\mohal\anaconda3\lib\site-packages (from missingno) (1.11.4)
Requirement already satisfied: seaborn in c:\users\mohal\anaconda3\lib\site-packages (from missingno) (0.12.2)
Requirement already satisfied: contourpy>=1.0.1 in c:\users\mohal\anaconda3\lib\site-packages (from matplotlib->
missingno) (1.2.0)
Requirement already satisfied: cycler>=0.10 in c:\users\mohal\anaconda3\lib\site-packages (from matplotlib->miss
ingno) (0.11.0)
Requirement already satisfied: fonttools>=4.22.0 in c:\users\mohal\anaconda3\lib\site-packages (from matplotlib-
>missingno) (4.25.0)
Requirement already satisfied: kiwisolver>=1.0.1 in c:\users\mohal\anaconda3\lib\site-packages (from matplotlib-
>missingno) (1.4.4)
Requirement already satisfied: packaging>=20.0 in c:\users\mohal\anaconda3\lib\site-packages (from matplotlib->m
issingno) (23.1)
Requirement already satisfied: pillow>=6.2.0 in c:\users\mohal\anaconda3\lib\site-packages (from matplotlib->mis
singno) (10.2.0)
Requirement already satisfied: pyparsing>=2.3.1 in c:\users\mohal\anaconda3\lib\site-packages (from matplotlib->
missingno) (3.0.9)
Requirement already satisfied: python-dateutil>=2.7 in c:\users\mohal\anaconda3\lib\site-packages (from matplotl
ib->missingno) (2.8.2)
Requirement already satisfied: pandas>=0.25 in c:\users\mohal\anaconda3\lib\site-packages (from seaborn->missing
no) (2.1.4)
Requirement already satisfied: pytz>=2020.1 in c:\users\mohal\anaconda3\lib\site-packages (from pandas>=0.25->se
aborn->missingno) (2023.3.post1)
Requirement already satisfied: tzdata>=2022.1 in c:\users\mohal\anaconda3\lib\site-packages (from pandas>=0.25->
seaborn->missingno) (2023.3)
Requirement already satisfied: six>=1.5 in c:\users\mohal\anaconda3\lib\site-packages (from python-dateutil>=2.7
->matplotlib->missingno) (1.16.0)
Note: you may need to restart the kernel to use updated packages.
```

In [1]:
```python
# Libraries for data loading, data manipulation and data visulisation
#import *
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
%matplotlib inline
import seaborn as sns
import plotly.express as px
import plotly.graph_objects as go
from statsmodels.graphics.correlation import plot_corr
import missingno as msno


# Libraries for data preparation and model building
#import *
from sklearn.linear_model import LinearRegression
from sklearn.model_selection import train_test_split
from sklearn import metrics
from sklearn.tree import DecisionTreeRegressor
from sklearn.metrics import accuracy_score
from sklearn.feature_selection import SelectKBest
from sklearn.feature_selection import chi2
from sklearn.metrics import mean_squared_error as MSE
from sklearn.ensemble import RandomForestRegressor
from sklearn.metrics import r2_score
from sklearn.preprocessing import StandardScaler
from sklearn.svm import SVR
import math
from statsmodels.graphics.correlation import plot_corr
import statsmodels.formula.api as sm
from statsmodels.formula.api import ols
from scipy.stats import pearsonr
from sklearn.linear_model import Lasso
from sklearn.linear_model import Ridge
from datetime import date
# Setting global constants to ensure notebook results are reproducible
PARAMETER_CONSTANT = 42
```

2. Loading the Data

Back to Table of Contents

⚡ Description: Loading the data ⚡

In this section you are required to load the data from the `df_train` file into a DataFrame.

In [2]:
```python
# load the data
```

```
train_data = pd.read_csv('df_train.csv') # load the train data
test_data = pd.read_csv('df_test.csv')  # load the test data
```

In [3]:
```
#overview of the dataset
print(f' There are {train_data.shape[0]} rows and {train_data.shape[1]} columns')
train_data.head(3)
```

```
There are 8763 rows and 49 columns
```

Out[3]:

	Unnamed: 0	time	Madrid_wind_speed	Valencia_wind_deg	Bilbao_rain_1h	Valencia_wind_speed	Seville_humidity	Madrid_hur
0	0	2015-01-01 03:00:00	0.666667	level_5	0.0	0.666667	74.333333	64.0(
1	1	2015-01-01 06:00:00	0.333333	level_10	0.0	1.666667	78.333333	64.6(
2	2	2015-01-01 09:00:00	1.000000	level_9	0.0	1.000000	71.333333	64.3

3 rows × 49 columns

3. Exploratory Data Analysis (EDA)

Back to Table of Contents

⚡ Description: Exploratory data analysis ⚡

In this section, you are required to perform an in-depth analysis of all the variables in the DataFrame.

In [5]:
```
#Reading the train data

train_data.head(5)
```

Out[5]:

	Unnamed: 0	time	Madrid_wind_speed	Valencia_wind_deg	Bilbao_rain_1h	Valencia_wind_speed	Seville_humidity	Madrid_hur
0	0	2015-01-01 03:00:00	0.666667	level_5	0.0	0.666667	74.333333	64.0(
1	1	2015-01-01 06:00:00	0.333333	level_10	0.0	1.666667	78.333333	64.6(
2	2	2015-01-01 09:00:00	1.000000	level_9	0.0	1.000000	71.333333	64.3
3	3	2015-01-01 12:00:00	1.000000	level_8	0.0	1.000000	65.333333	56.3
4	4	2015-01-01 15:00:00	1.000000	level_7	0.0	1.000000	59.000000	57.0(

5 rows × 49 columns

In [6]:
```
#Reading the data in a transposed form to view all the available features

train_data.head(5).T
```

Out[6]:

	0	1	2	3	4
Unnamed: 0	0	1	2	3	4
time	2015-01-01 03:00:00	2015-01-01 06:00:00	2015-01-01 09:00:00	2015-01-01 12:00:00	2015-01-01 15:00:00
Madrid_wind_speed	0.666667	0.333333	1.0	1.0	1.0
Valencia_wind_deg	level_5	level_10	level_9	level_8	level_7
Bilbao_rain_1h	0.0	0.0	0.0	0.0	0.0
Valencia_wind_speed	0.666667	1.666667	1.0	1.0	1.0
Seville_humidity	74.333333	78.333333	71.333333	65.333333	59.0
Madrid_humidity	64.0	64.666667	64.333333	56.333333	57.0
Bilbao_clouds_all	0.0	0.0	0.0	0.0	2.0
Bilbao_wind_speed	1.0	1.0	1.0	1.0	0.333333
Seville_clouds_all	0.0	0.0	0.0	0.0	0.0
Bilbao_wind_deg	223.333333	221.0	214.333333	199.666667	185.0
Barcelona_wind_speed	6.333333	4.0	2.0	2.333333	4.333333
Barcelona_wind_deg	42.666667	139.0	326.0	273.0	260.0
Madrid_clouds_all	0.0	0.0	0.0	0.0	0.0
Seville_wind_speed	3.333333	3.333333	2.666667	4.0	3.0
Barcelona_rain_1h	0.0	0.0	0.0	0.0	0.0
Seville_pressure	sp25	sp25	sp25	sp25	sp25
Seville_rain_1h	0.0	0.0	0.0	0.0	0.0
Bilbao_snow_3h	0.0	0.0	0.0	0.0	0.0
Barcelona_pressure	1036.333333	1037.333333	1038.0	1037.0	1035.0
Seville_rain_3h	0.0	0.0	0.0	0.0	0.0
Madrid_rain_1h	0.0	0.0	0.0	0.0	0.0
Barcelona_rain_3h	0.0	0.0	0.0	0.0	0.0
Valencia_snow_3h	0.0	0.0	0.0	0.0	0.0
Madrid_weather_id	800.0	800.0	800.0	800.0	800.0
Barcelona_weather_id	800.0	800.0	800.0	800.0	800.0
Bilbao_pressure	1035.0	1035.666667	1036.0	1036.0	1035.333333
Seville_weather_id	800.0	800.0	800.0	800.0	800.0
Valencia_pressure	1002.666667	1004.333333	1005.333333	1009.0	NaN
Seville_temp_max	274.254667	274.945	278.792	285.394	285.513719
Madrid_pressure	971.333333	972.666667	974.0	994.666667	1035.333333
Valencia_temp_max	269.888	271.728333	278.008667	284.899552	283.015115
Valencia_temp	269.888	271.728333	278.008667	284.899552	283.015115
Bilbao_weather_id	800.0	800.0	800.0	800.0	800.0
Seville_temp	274.254667	274.945	278.792	285.394	285.513719
Valencia_humidity	75.666667	71.0	65.666667	54.0	58.333333
Valencia_temp_min	269.888	271.728333	278.008667	284.899552	283.015115
Barcelona_temp_max	281.013	280.561667	281.583667	283.434104	284.213167
Madrid_temp_max	265.938	266.386667	272.708667	281.895219	280.678437
Barcelona_temp	281.013	280.561667	281.583667	283.434104	284.213167
Bilbao_temp_min	269.338615	270.376	275.027229	281.135063	282.252063
Bilbao_temp	269.338615	270.376	275.027229	281.135063	282.252063
Barcelona_temp_min	281.013	280.561667	281.583667	283.434104	284.213167
Bilbao_temp_max	269.338615	270.376	275.027229	281.135063	282.252063
Seville_temp_min	274.254667	274.945	278.792	285.394	285.513719
Madrid_temp	265.938	266.386667	272.708667	281.895219	280.678437
Madrid_temp_min	265.938	266.386667	272.708667	281.895219	280.678437
load_shortfall_3h	6715.666667	4171.666667	4274.666667	5075.666667	6620.666667

In [7]:
```
#Shape of the data
train_data.shape
```

Out[7]: (8763, 49)

What can we observe from the table above?

1.The Table has 8763 and 49 column.

2.There is a column with the name Unnamed which is actually insignificant and therefore needs to be dropped.

In [8]:
```
#Checking for data type of the columns
train_data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 8763 entries, 0 to 8762
Data columns (total 49 columns):
 #   Column                Non-Null Count  Dtype  
---  ------                --------------  -----  
 0   Unnamed: 0            8763 non-null   int64  
 1   time                  8763 non-null   object 
 2   Madrid_wind_speed     8763 non-null   float64
 3   Valencia_wind_deg     8763 non-null   object 
 4   Bilbao_rain_1h        8763 non-null   float64
 5   Valencia_wind_speed   8763 non-null   float64
 6   Seville_humidity      8763 non-null   float64
 7   Madrid_humidity       8763 non-null   float64
 8   Bilbao_clouds_all     8763 non-null   float64
 9   Bilbao_wind_speed     8763 non-null   float64
 10  Seville_clouds_all    8763 non-null   float64
 11  Bilbao_wind_deg       8763 non-null   float64
 12  Barcelona_wind_speed  8763 non-null   float64
 13  Barcelona_wind_deg    8763 non-null   float64
 14  Madrid_clouds_all     8763 non-null   float64
 15  Seville_wind_speed    8763 non-null   float64
 16  Barcelona_rain_1h     8763 non-null   float64
 17  Seville_pressure      8763 non-null   object 
 18  Seville_rain_1h       8763 non-null   float64
 19  Bilbao_snow_3h        8763 non-null   float64
 20  Barcelona_pressure    8763 non-null   float64
 21  Seville_rain_3h       8763 non-null   float64
 22  Madrid_rain_1h        8763 non-null   float64
 23  Barcelona_rain_3h     8763 non-null   float64
 24  Valencia_snow_3h      8763 non-null   float64
 25  Madrid_weather_id     8763 non-null   float64
 26  Barcelona_weather_id  8763 non-null   float64
 27  Bilbao_pressure       8763 non-null   float64
 28  Seville_weather_id    8763 non-null   float64
 29  Valencia_pressure     6695 non-null   float64
 30  Seville_temp_max      8763 non-null   float64
 31  Madrid_pressure       8763 non-null   float64
 32  Valencia_temp_max     8763 non-null   float64
 33  Valencia_temp         8763 non-null   float64
 34  Bilbao_weather_id     8763 non-null   float64
 35  Seville_temp          8763 non-null   float64
 36  Valencia_humidity     8763 non-null   float64
 37  Valencia_temp_min     8763 non-null   float64
 38  Barcelona_temp_max    8763 non-null   float64
 39  Madrid_temp_max       8763 non-null   float64
 40  Barcelona_temp        8763 non-null   float64
 41  Bilbao_temp_min       8763 non-null   float64
 42  Bilbao_temp           8763 non-null   float64
 43  Barcelona_temp_min    8763 non-null   float64
 44  Bilbao_temp_max       8763 non-null   float64
 45  Seville_temp_min      8763 non-null   float64
 46  Madrid_temp           8763 non-null   float64
 47  Madrid_temp_min       8763 non-null   float64
 48  load_shortfall_3h     8763 non-null   float64
dtypes: float64(45), int64(1), object(3)
memory usage: 3.3+ MB
```

Investigating null values

From the table above we can see that the data types of some columns(time,Valencia_wind_speed and Seville_pressure) are of object type which is categorical and they need to be of numerical data type.

In [9]:
```
#Checking for any occurrence of null values
train_data.isnull().sum()
```

Out[9]:
```
Unnamed: 0                0
time                      0
Madrid_wind_speed         0
Valencia_wind_deg         0
Bilbao_rain_1h            0
Valencia_wind_speed       0
Seville_humidity          0
Madrid_humidity           0
Bilbao_clouds_all         0
Bilbao_wind_speed         0
Seville_clouds_all        0
Bilbao_wind_deg           0
Barcelona_wind_speed      0
Barcelona_wind_deg        0
Madrid_clouds_all         0
Seville_wind_speed        0
Barcelona_rain_1h         0
Seville_pressure          0
Seville_rain_1h           0
Bilbao_snow_3h            0
Barcelona_pressure        0
Seville_rain_3h           0
Madrid_rain_1h            0
Barcelona_rain_3h         0
Valencia_snow_3h          0
Madrid_weather_id         0
Barcelona_weather_id      0
Bilbao_pressure           0
Seville_weather_id        0
Valencia_pressure      2068
Seville_temp_max          0
Madrid_pressure           0
Valencia_temp_max         0
Valencia_temp             0
Bilbao_weather_id         0
Seville_temp              0
Valencia_humidity         0
Valencia_temp_min         0
Barcelona_temp_max        0
Madrid_temp_max           0
Barcelona_temp            0
Bilbao_temp_min           0
Bilbao_temp               0
Barcelona_temp_min        0
Bilbao_temp_max           0
Seville_temp_min          0
Madrid_temp               0
Madrid_temp_min           0
load_shortfall_3h         0
dtype: int64
```

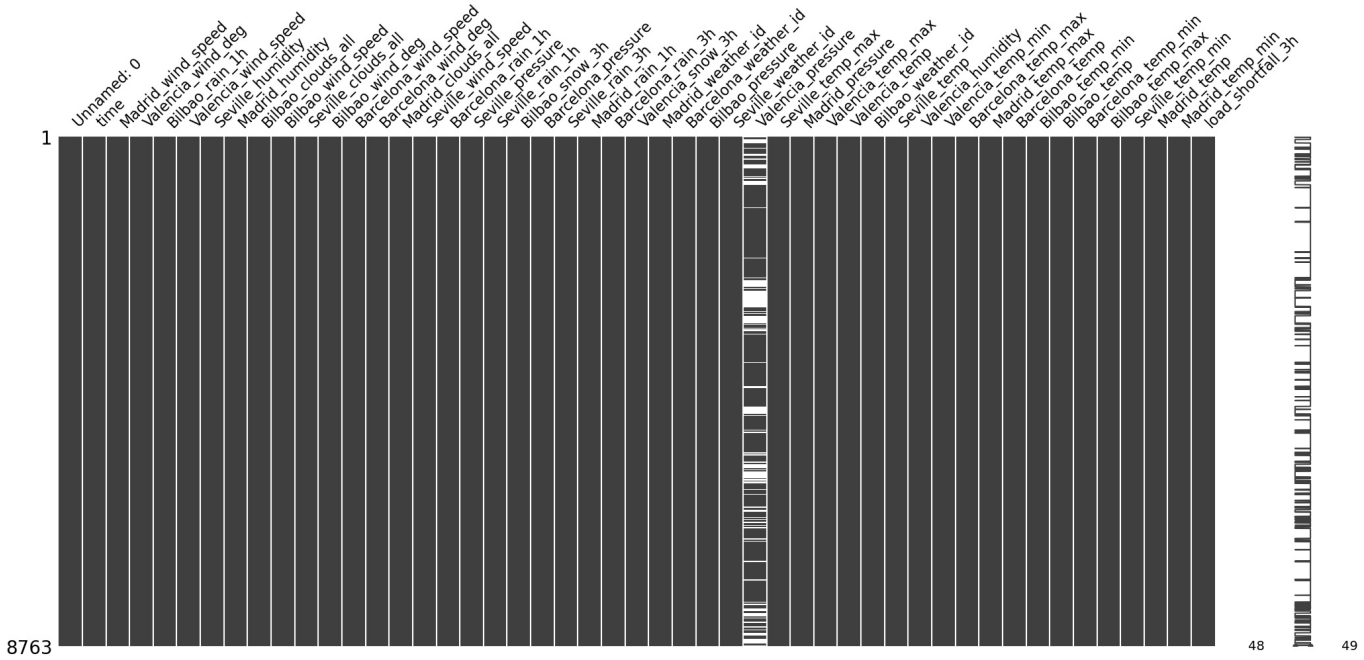
We can observe from the above the above table that the column('Seville_pressure') has 2068 null values present.

It can also be seen by visualising the missing data,which makes it very clear that the 'Seville_pressure' feature has large amounts of missing data.

In [10]:
```
# investigate missing data

msno.matrix(train_data)
```

Out[10]:
```
<Axes: >
```



In [11]:
```
# look at data statistics of this column

print('Mode')
print(train_data['Valencia_pressure'].mode())
print('Mean')
print(train_data['Valencia_pressure'].mean())
print('Median')
print(train_data['Valencia_pressure'].median())
```

```
Mode
0    1018.0
Name: Valencia_pressure, dtype: float64
Mean
1012.0514065222798
Median
1015.0
```

Now we can perform descriptive statistics of our data below

In [12]:
```
#Look at data statistics
statistics = train_data.describe()
statistics.T
```

Out[12]:

	count	mean	std	min	25%	50%	75%	max
Unnamed: 0	8763.0	4381.000000	2529.804538	0.000000	2190.500000	4381.000000	6571.500000	8.762000e+03
Madrid_wind_speed	8763.0	2.425729	1.850371	0.000000	1.000000	2.000000	3.333333	1.300000e+01
Bilbao_rain_1h	8763.0	0.135753	0.374901	0.000000	0.000000	0.000000	0.100000	3.000000e+00
Valencia_wind_speed	8763.0	2.586272	2.411190	0.000000	1.000000	1.666667	3.666667	5.200000e+01
Seville_humidity	8763.0	62.658793	22.621226	8.333333	44.333333	65.666667	82.000000	1.000000e+02
Madrid_humidity	8763.0	57.414717	24.335396	6.333333	36.333333	58.000000	78.666667	1.000000e+02
Bilbao_clouds_all	8763.0	43.469132	32.551044	0.000000	10.000000	45.000000	75.000000	1.000000e+02
Bilbao_wind_speed	8763.0	1.850356	1.695888	0.000000	0.666667	1.000000	2.666667	1.266667e+01
Seville_clouds_all	8763.0	13.714748	24.272482	0.000000	0.000000	0.000000	20.000000	9.733333e+01
Bilbao_wind_deg	8763.0	158.957511	102.056299	0.000000	73.333333	147.000000	234.000000	3.593333e+02
Barcelona_wind_speed	8763.0	2.870497	1.792197	0.000000	1.666667	2.666667	4.000000	1.266667e+01
Barcelona_wind_deg	8763.0	190.544848	89.077337	0.000000	118.166667	200.000000	260.000000	3.600000e+02
Madrid_clouds_all	8763.0	19.473392	28.053660	0.000000	0.000000	0.000000	33.333333	1.000000e+02
Seville_wind_speed	8763.0	2.425045	1.672895	0.000000	1.000000	2.000000	3.333333	1.166667e+01
Barcelona_rain_1h	8763.0	0.128906	0.634730	0.000000	0.000000	0.000000	0.000000	1.200000e+01
Seville_rain_1h	8763.0	0.039439	0.175857	0.000000	0.000000	0.000000	0.000000	3.000000e+00
Bilbao_snow_3h	8763.0	0.031912	0.557264	0.000000	0.000000	0.000000	0.000000	2.130000e+01
Barcelona_pressure	8763.0	1377.964605	14073.140990	670.666667	1014.000000	1018.000000	1022.000000	1.001411e+06
Seville_rain_3h	8763.0	0.000243	0.003660	0.000000	0.000000	0.000000	0.000000	9.333333e-02
Madrid_rain_1h	8763.0	0.037818	0.152639	0.000000	0.000000	0.000000	0.000000	3.000000e+00
Barcelona_rain_3h	8763.0	0.000439	0.003994	0.000000	0.000000	0.000000	0.000000	9.300000e-02
Valencia_snow_3h	8763.0	0.000205	0.011866	0.000000	0.000000	0.000000	0.000000	7.916667e-01
Madrid_weather_id	8763.0	773.527594	77.313315	211.000000	800.000000	800.000000	800.666667	8.040000e+02
Barcelona_weather_id	8763.0	765.979687	88.142235	200.666667	800.000000	800.333333	801.000000	8.040000e+02
Bilbao_pressure	8763.0	1017.739549	10.046124	971.333333	1013.000000	1019.000000	1024.000000	1.042000e+03
Seville_weather_id	8763.0	774.658818	71.940009	200.000000	800.000000	800.000000	800.000000	8.040000e+02
Valencia_pressure	6695.0	1012.051407	9.506214	972.666667	1010.333333	1015.000000	1018.000000	1.021667e+03
Seville_temp_max	8763.0	297.479527	8.875812	272.063000	291.312750	297.101667	304.150000	3.204833e+02
Madrid_pressure	8763.0	1010.316920	22.198555	927.666667	1012.333333	1017.333333	1022.000000	1.038000e+03
Valencia_temp_max	8763.0	291.337233	7.565692	269.888000	285.550167	291.037000	297.248333	3.142633e+02
Valencia_temp	8763.0	290.592152	7.162274	269.888000	285.150000	290.176667	296.056667	3.104267e+02
Bilbao_weather_id	8763.0	724.722362	115.846537	207.333333	700.333333	800.000000	801.666667	8.040000e+02
Seville_temp	8763.0	293.978903	7.920986	272.063000	288.282917	293.323333	299.620333	3.149767e+02
Valencia_humidity	8763.0	65.247727	19.262322	10.333333	51.333333	67.000000	81.333333	1.000000e+02
Valencia_temp_min	8763.0	289.867648	6.907402	269.888000	284.783333	289.550000	294.820000	3.102720e+02
Barcelona_temp_max	8763.0	291.157644	7.273538	272.150000	285.483333	290.150000	296.855000	3.140767e+02
Madrid_temp_max	8763.0	289.540309	9.752047	264.983333	282.150000	288.116177	296.816667	3.144833e+02
Barcelona_temp	8763.0	289.855459	6.528111	270.816667	284.973443	289.416667	294.909000	3.073167e+02
Bilbao_temp_min	8763.0	285.017973	6.705672	264.483333	280.085167	284.816667	289.816667	3.098167e+02
Bilbao_temp	8763.0	286.422929	6.818682	267.483333	281.374167	286.158333	291.034167	3.107100e+02
Barcelona_temp_min	8763.0	288.447422	6.102593	269.483333	284.150000	288.150000	292.966667	3.048167e+02
Bilbao_temp_max	8763.0	287.966027	7.105590	269.063000	282.836776	287.630000	292.483333	3.179667e+02
Seville_temp_min	8763.0	291.633356	8.178220	270.150000	285.816667	290.816667	297.150000	3.148167e+02
Madrid_temp	8763.0	288.419439	9.346796	264.983333	281.404281	287.053333	295.154667	3.131333e+02
Madrid_temp_min	8763.0	287.202203	9.206237	264.983333	280.299167	286.083333	293.884500	3.103833e+02
load_shortfall_3h	8763.0	10673.857612	5218.046404	-6618.000000	7390.333333	11114.666667	14498.166667	3.190400e+04

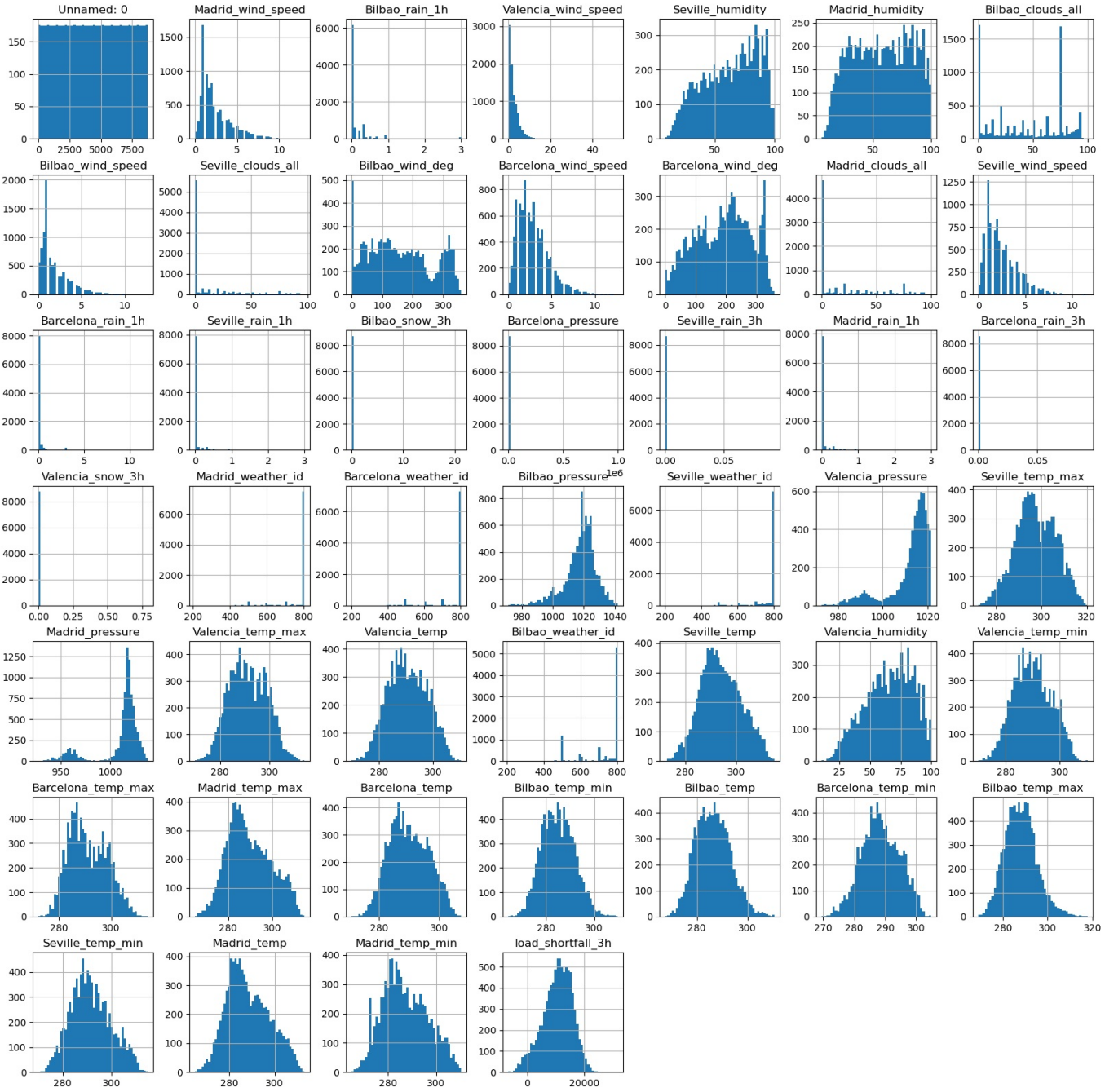
Looking at our data, we can infer the following:

1.We can tell from the Mean(average) that some of the columns such as Barcelona Pressure, Bilbao Pressure, Valencia Pressure etc. show very large values which is far from the range.

2.This can also be seen in their maximum values which would deduce the presence of outliers.

This can be confirmed by doing further analysis

In [4]:
```
# plotting all features to see the distribution of values
train_data.hist(bins=50, figsize=(20,20))
plt.show()
```



Now we can strore each city of spain into diffecrent variables to be used for feature distributions

In [5]:
```
#store each city into avariable with an equivalent name to the city

madrid = train_data[[col for col in train_data if col.startswith('Madrid')] + ['load_shortfall_3h']]
seville = train_data[[col for col in train_data if col.startswith('Seville')] + ['load_shortfall_3h']]
barcelona = train_data[[col for col in train_data if col.startswith('Barcelona')] + ['load_shortfall_3h']]
valencia = train_data[[col for col in train_data if col.startswith('Valencia')] + ['load_shortfall_3h']]
bilbao = train_data[[col for col in train_data if col.startswith('Bilbao')] + ['load_shortfall_3h']]
```

In [6]:
```
# plot relevant feature interactions
# Define a Function to Help Plot Values for reusability
def scatter_plot(data, row = 2, col = 5, color='blue'):
    fig, axs = plt.subplots(row,col, figsize=(14,6),)
    fig.subplots_adjust(hspace = 1.0, wspace=.4)
    axs = axs.ravel()
    plot = [col for col in data.columns if data[col].dtype == float and col != 'load_shortfall_3h']
    for index, column in enumerate(plot):
```

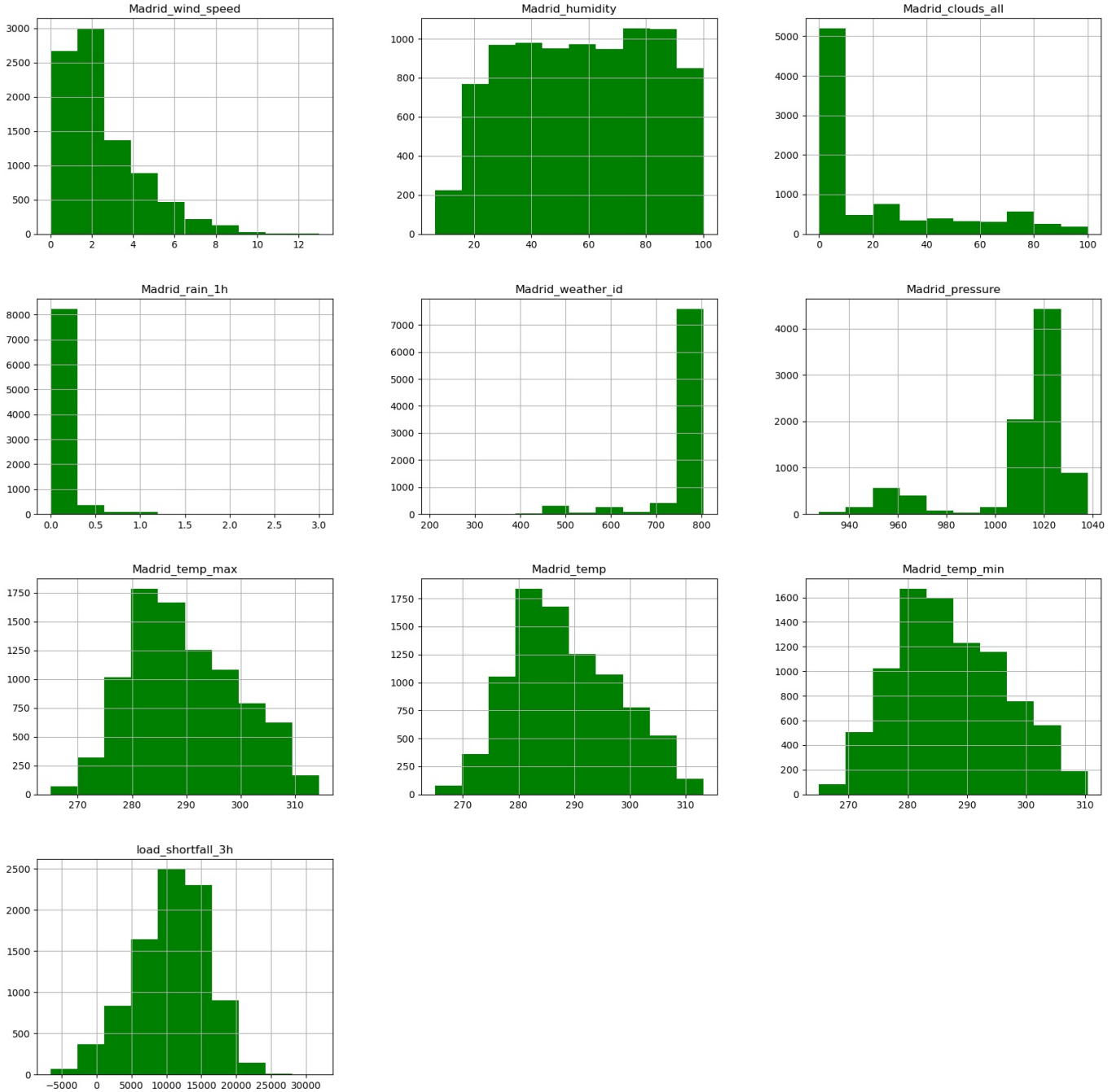
```
        axs[index-1].set_title("{}".format(column),fontsize=16)
        axs[index-1].scatter(x=data[column],y=data['load_shortfall_3h'],color=color,edgecolor='k')

    fig.tight_layout(pad=1)
```

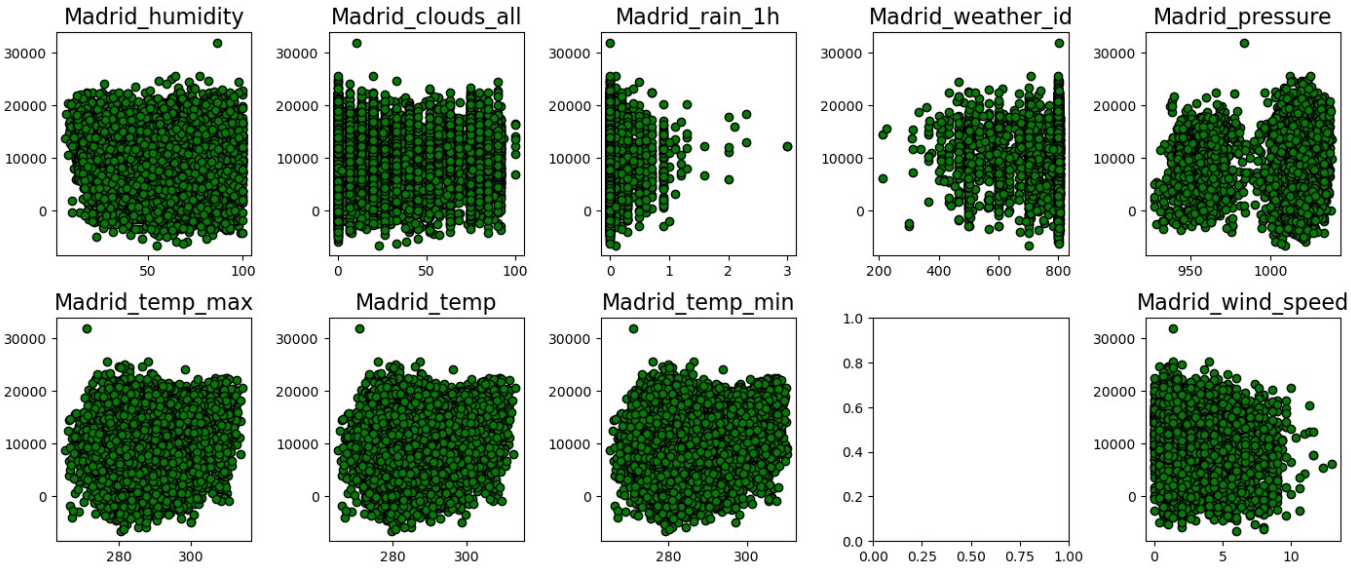
In [7]:
```
def plot_categorical(column, data):
    plt.figure(figsize=(15,10))
    sns.countplot(x=column, data = data, palette = 'Set2',
                  edgecolor=sns.color_palette("dark", 3),
                  order=data[column].value_counts().index[:30])
```

MADRID

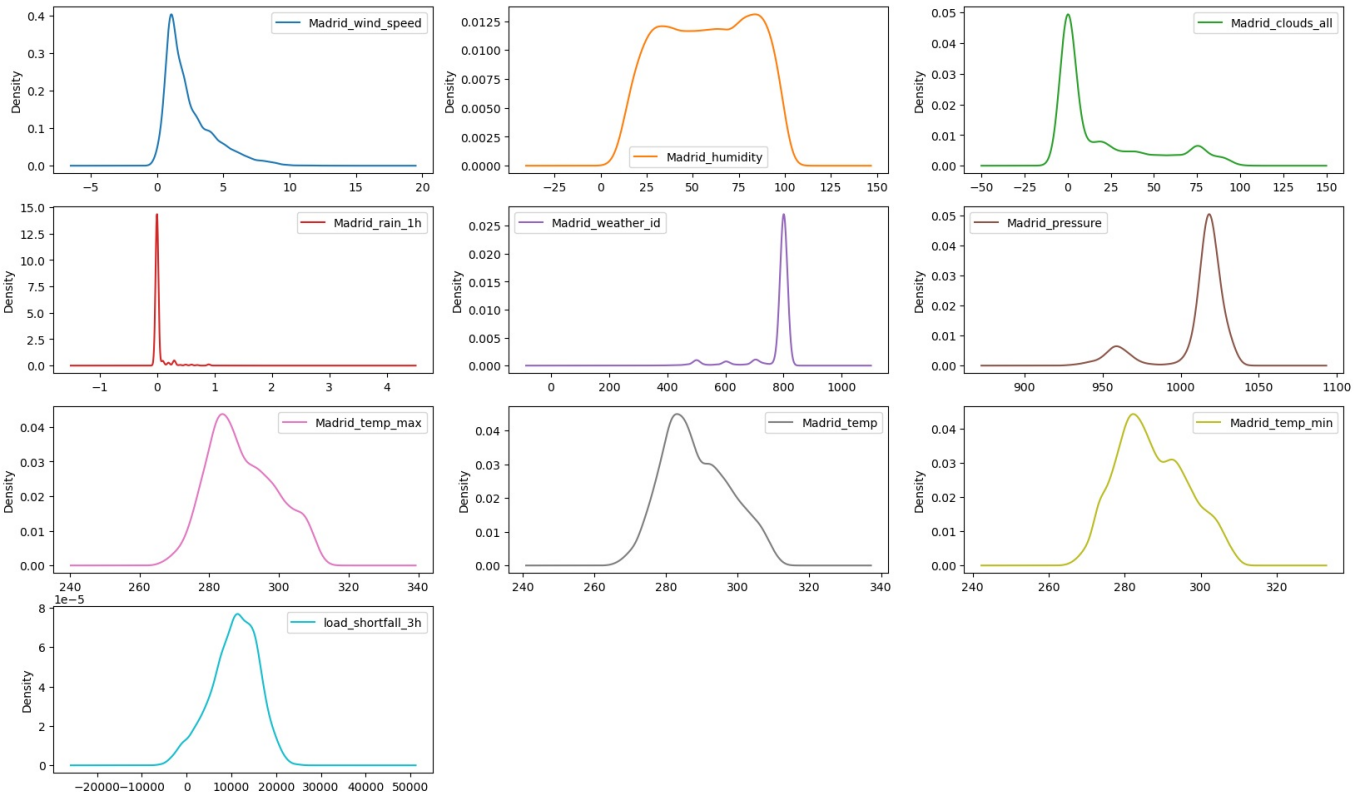
In [8]:
```
# have a look at feature distributions Using histogram Plots for the amdrid feature
madrid.hist(figsize=(20,20), color = 'green');
```



In [9]:
```
# Make Scatter Plots of Variables vs load_shortfall_3h for madrid
scatter_plot(madrid, color = 'green')
```

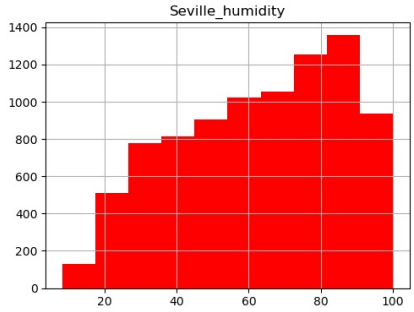


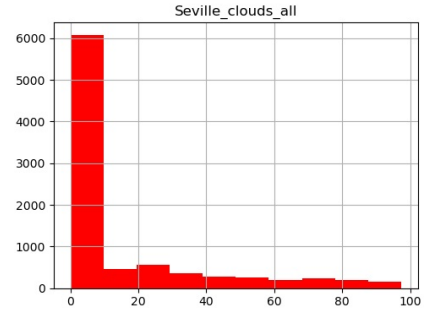
In [10]:
```
# have a look at feature distributions
madrid.plot(kind= 'density', subplots=True, layout=(13,3), sharex= False, figsize= (20,40))
plt.show()
```

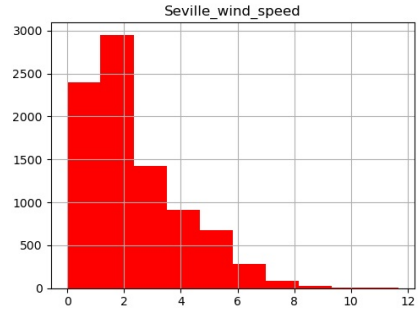


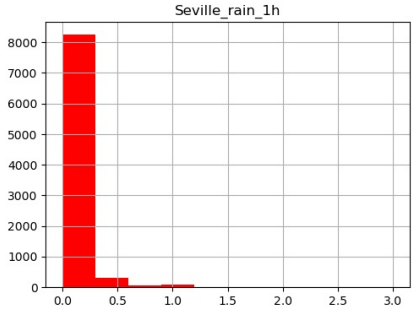
SEVILLE

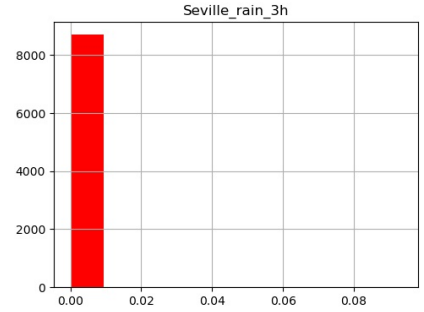
In [11]:
```
# have a look at feature distributions Using histogram Plots
seville.hist(figsize=(20,20), color = 'red');
```

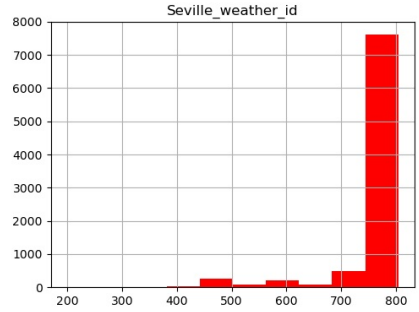


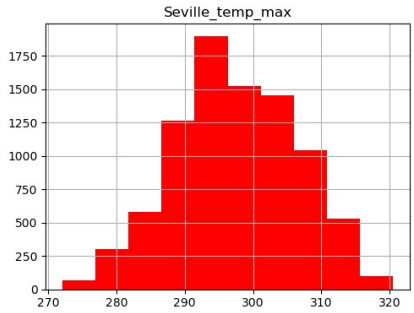


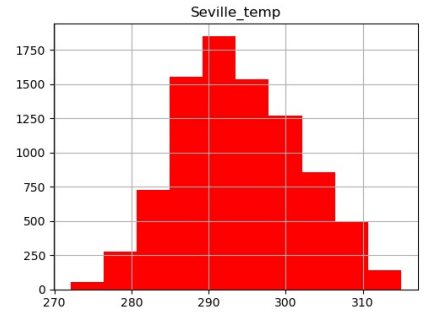


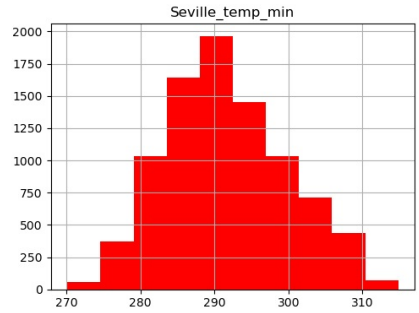


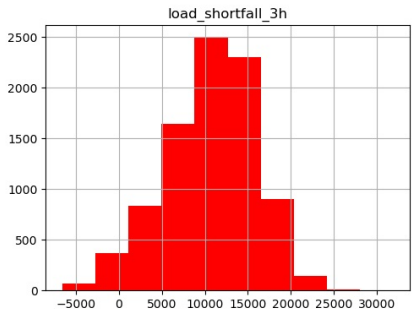




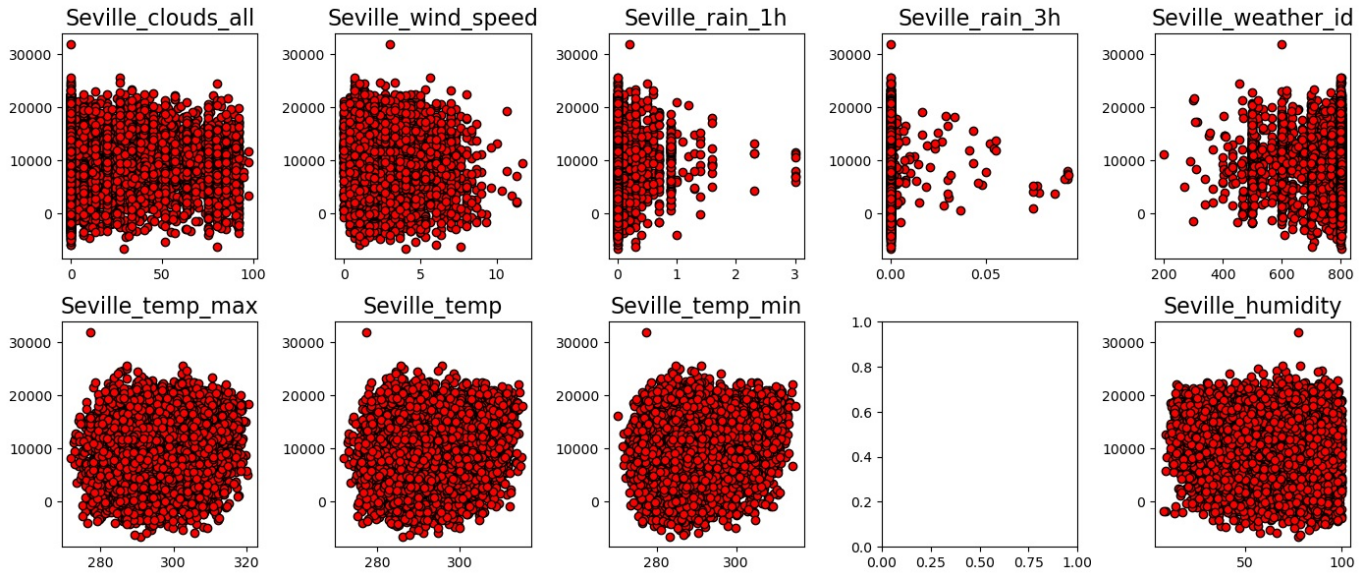




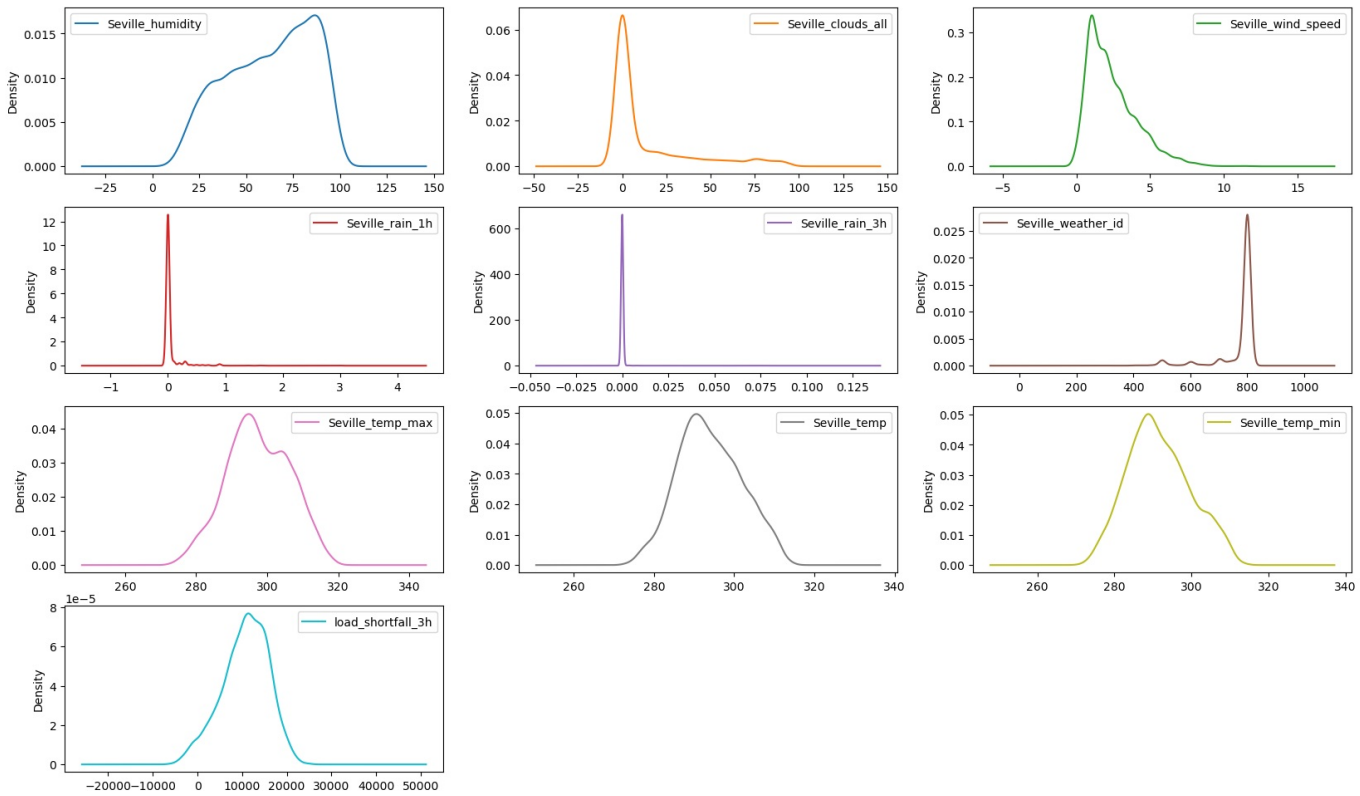




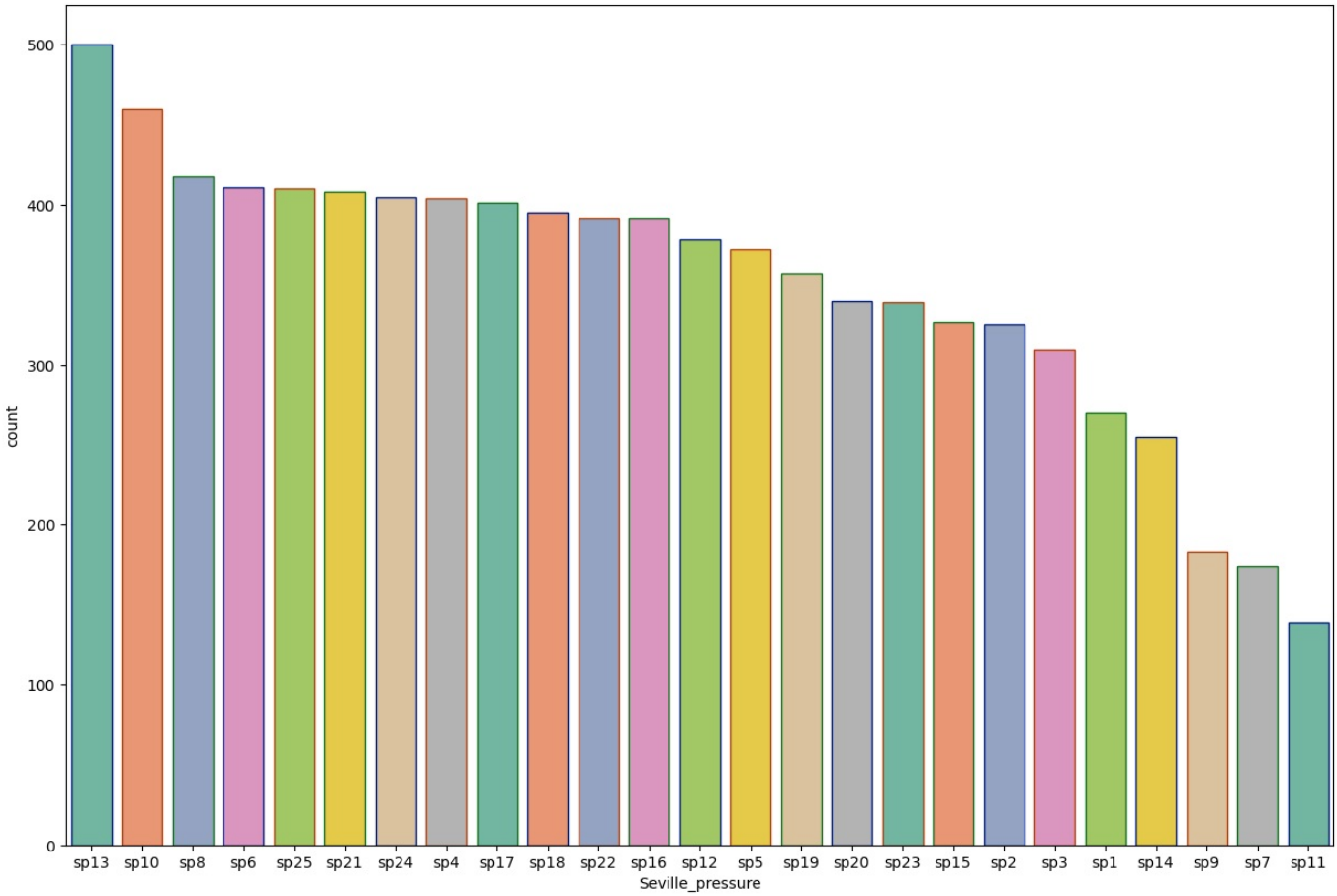
In [12]:
```
# Make Scatter Plots of Variables vs load_shortfall_3h
scatter_plot(seville, color = 'red')
```



In [13]:
```
# have a look at feature distributions
seville.plot(kind= 'density', subplots=True, layout=(13,3), sharex= False, figsize= (20,40))
plt.show()
```

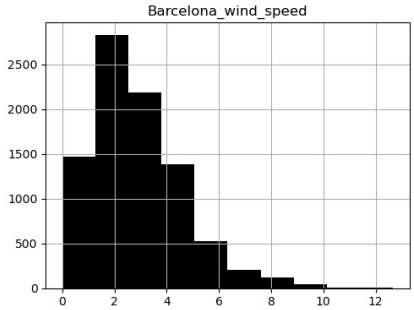


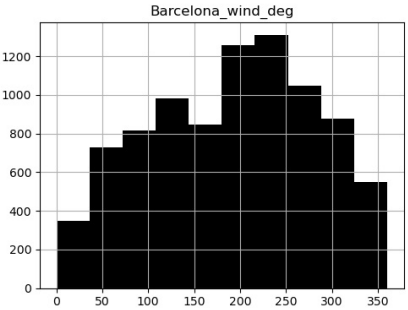
In [14]:
```
# Plot categroical variable to view distribution
plot_categorical('Seville_pressure', data= seville)
```

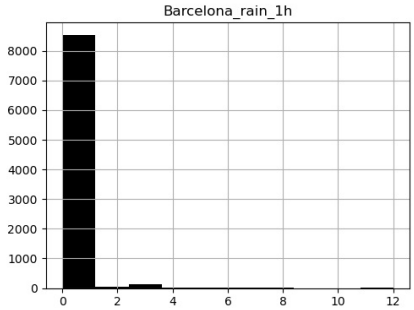


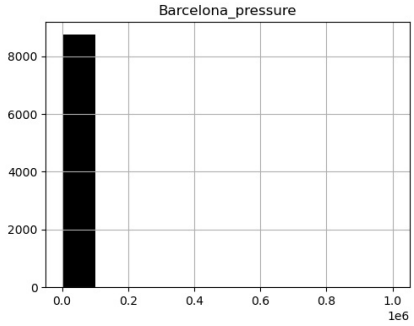
BARCELONA

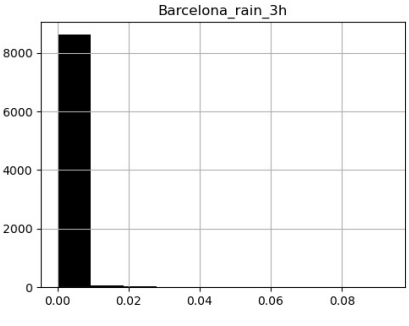
In [15]:
```
# have a look at feature distributions Using histogram Plots
barcelona.hist(figsize=(20,20), color = 'black');
```

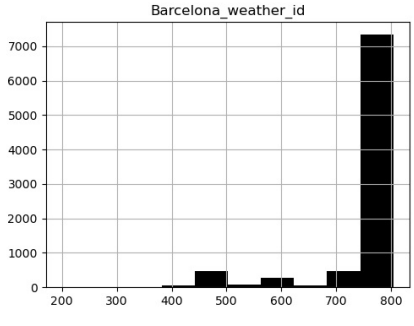


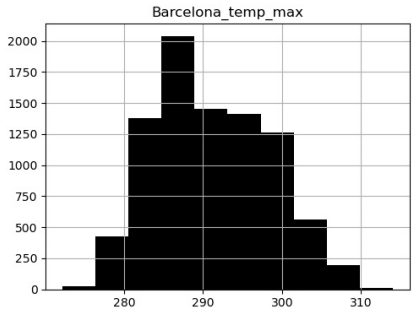


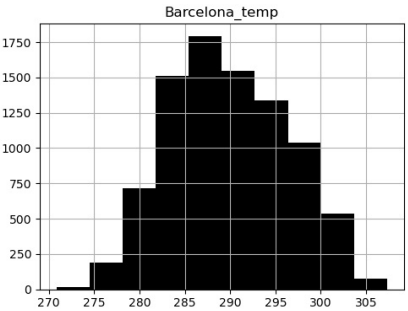


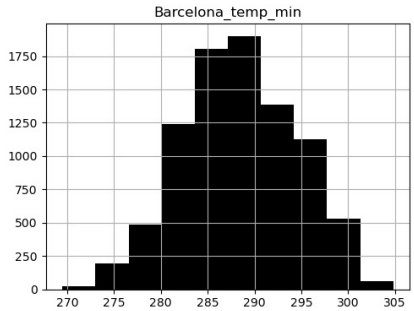


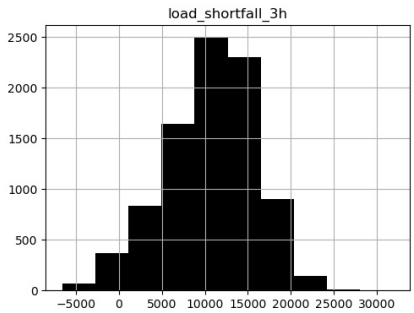




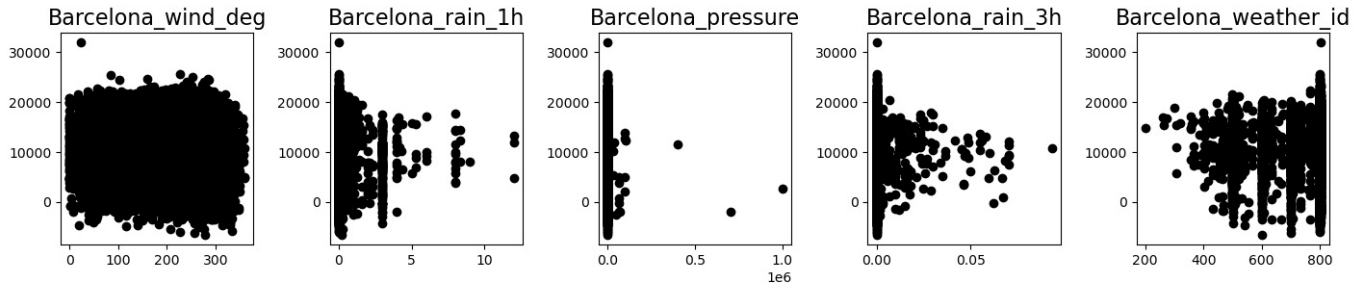


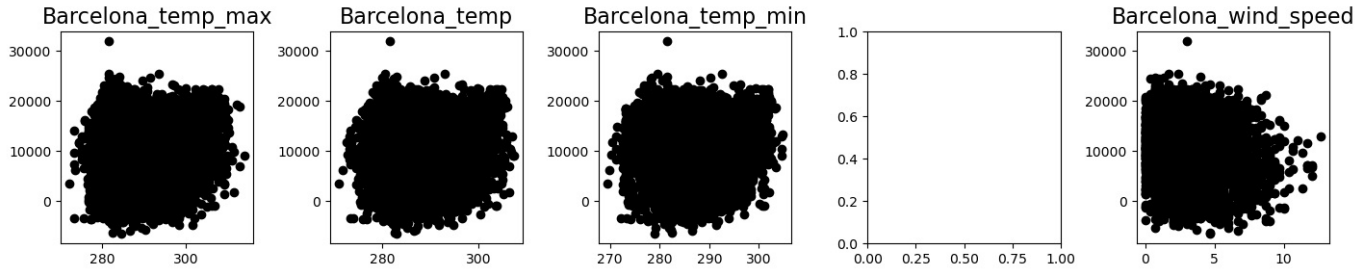




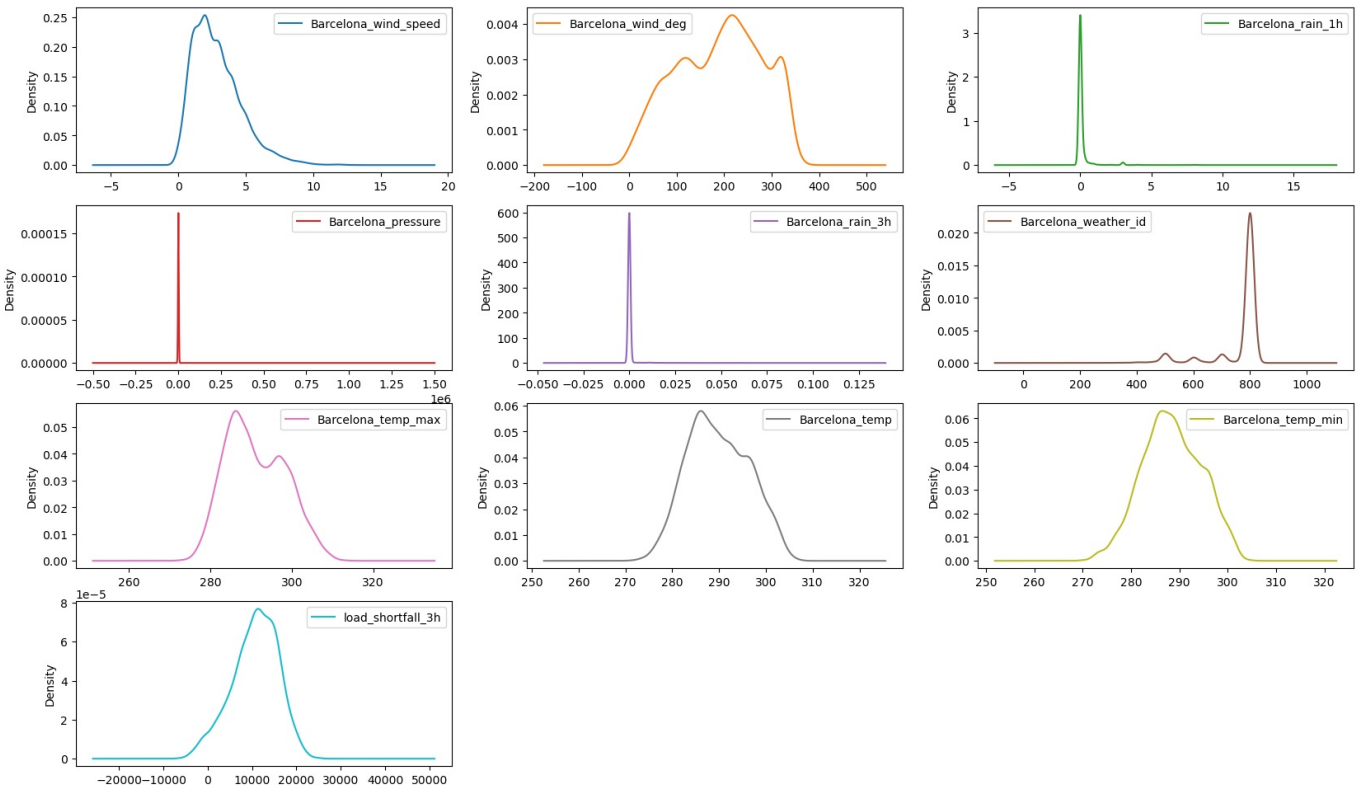


In [16]:
```
# Make Scatter Plots of Variables vs load_shortfall_3h
scatter_plot(barcelona, color = 'black')
```



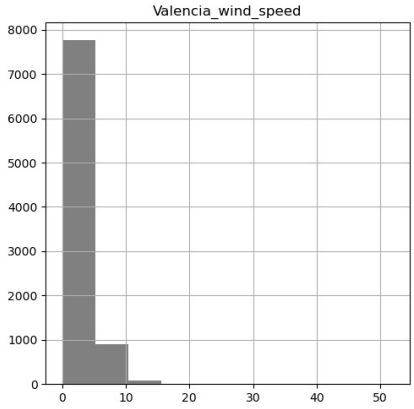


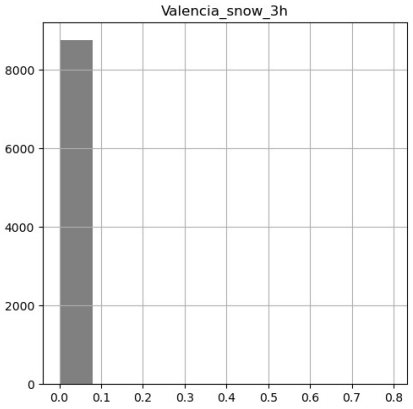
In [17]:
```
# have a look at feature distributions
barcelona.plot(kind= 'density', subplots=True, layout=(13,3), sharex= False, figsize= (20,40))
plt.show()
```

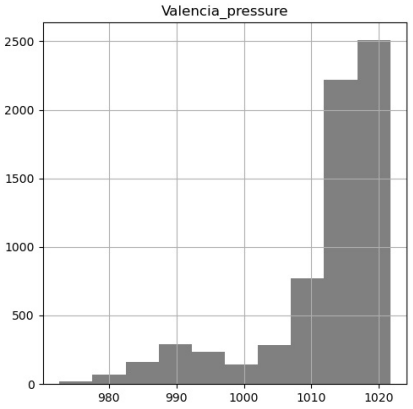


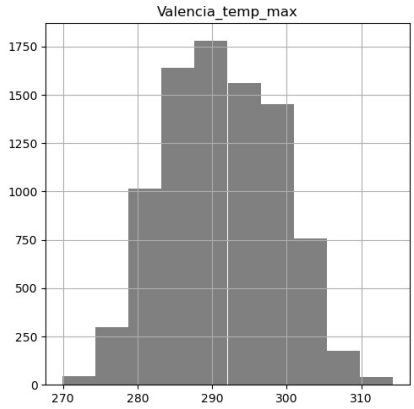
VALENCIA

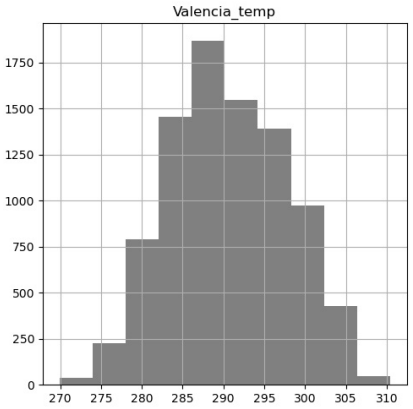
In [18]:
```
# have a look at feature distributions Using histogram Plots
valencia.hist(figsize=(20,20), color = 'grey');
```

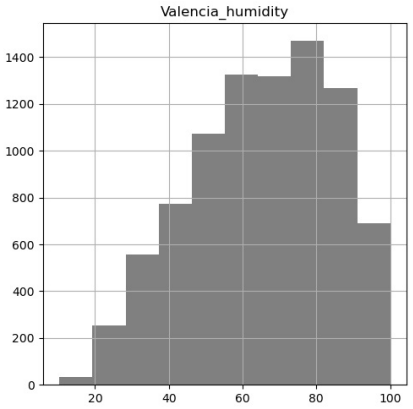


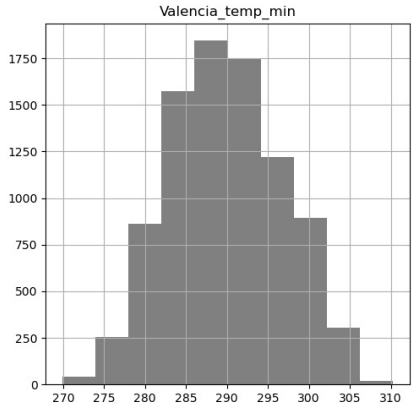


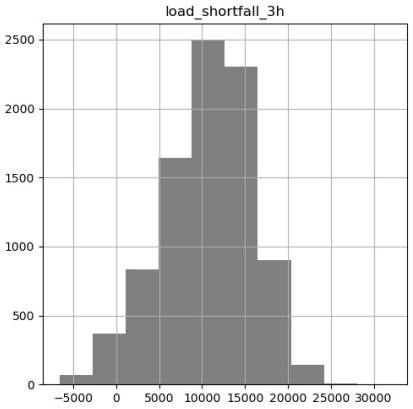




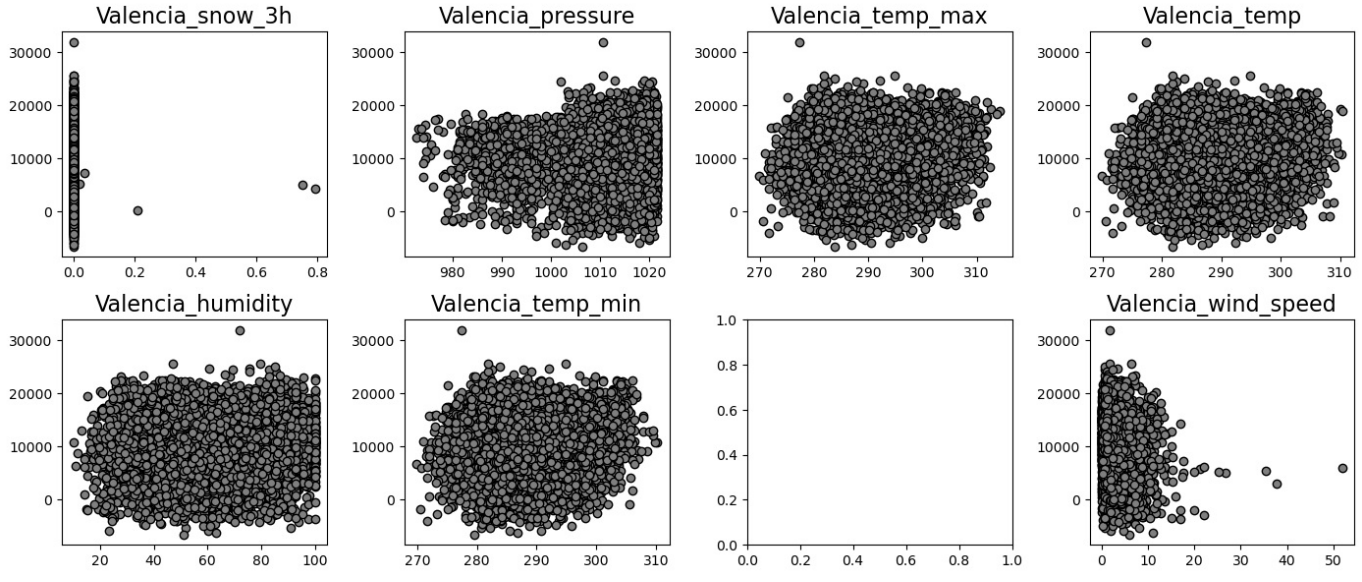




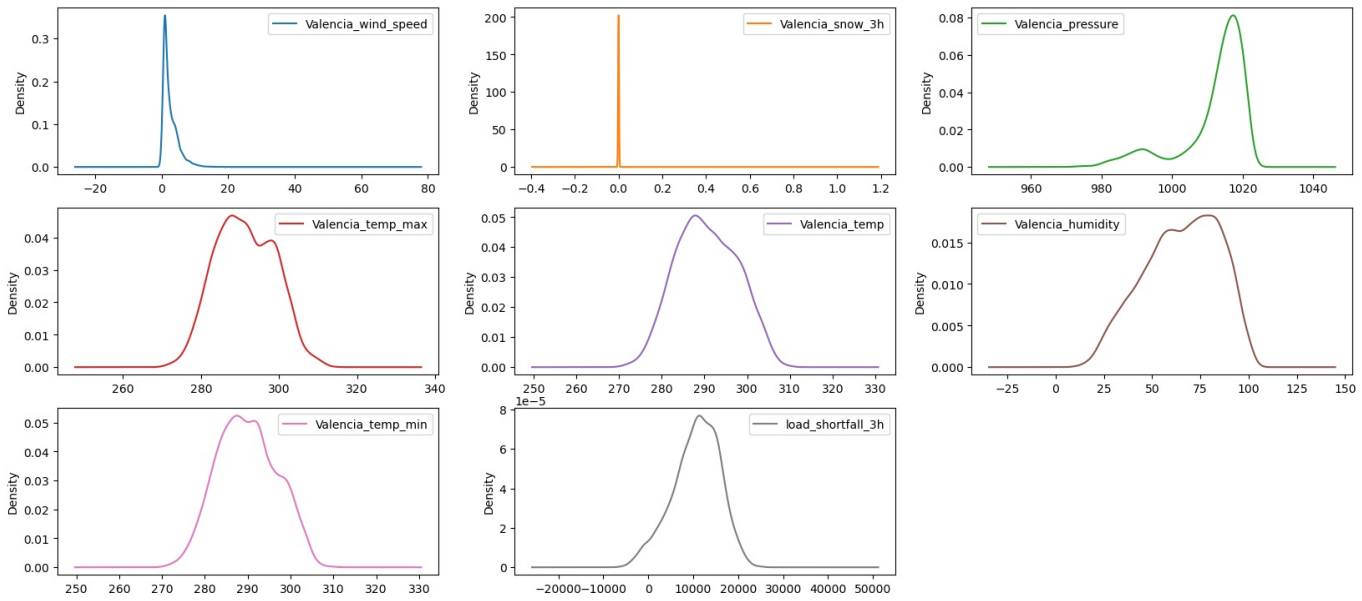




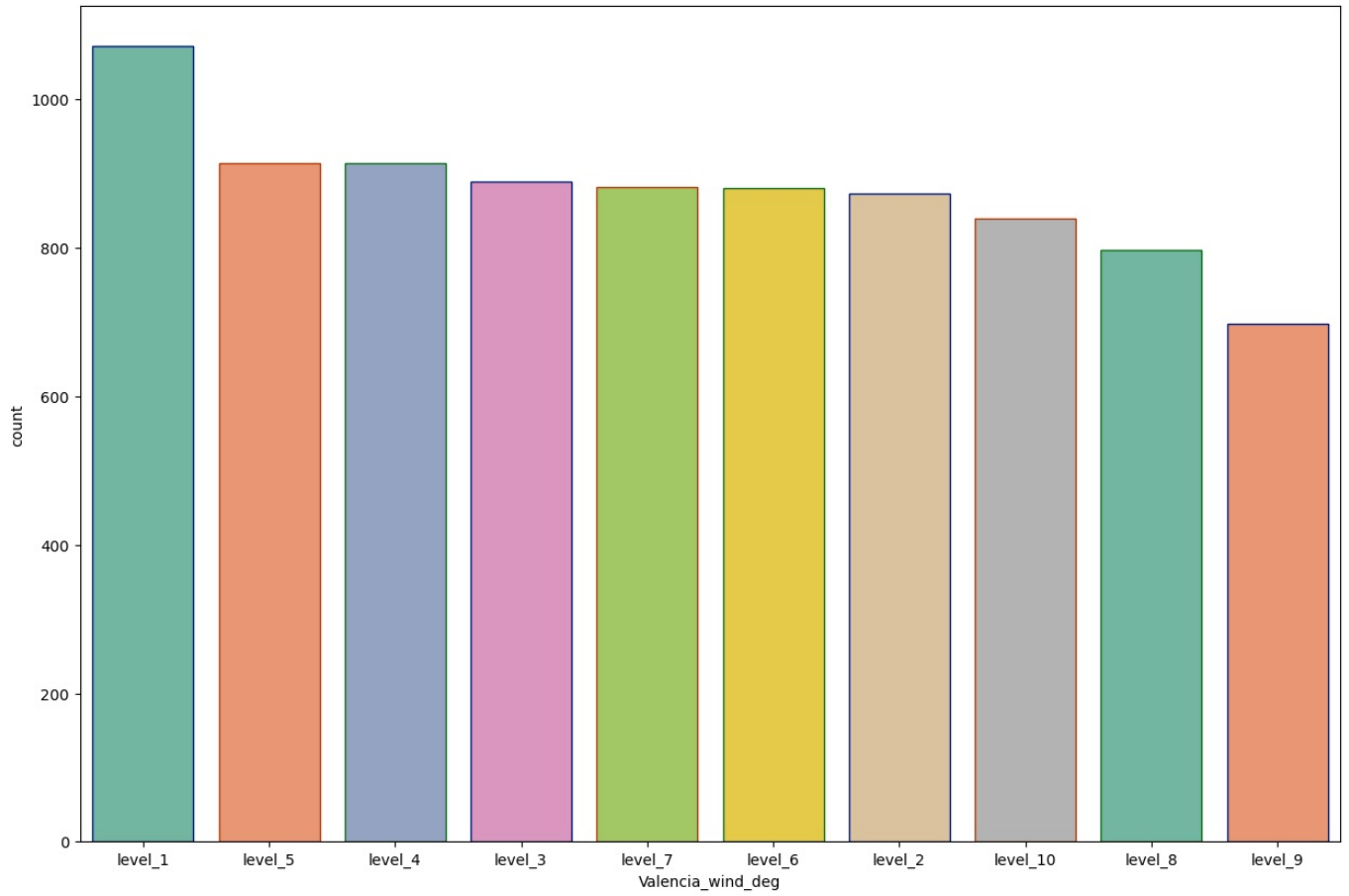
In [19]:
```
# Make Scatter Plots of Variables vs load_shortfall_3h
scatter_plot(valencia, color = 'grey', col = 4)
```



In [20]:
```
# have a look at feature distributions
valencia.plot(kind= 'density', subplots=True, layout=(13,3), sharex= False, figsize= (20,40))
plt.show()
```

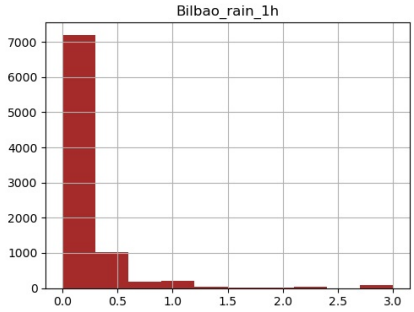


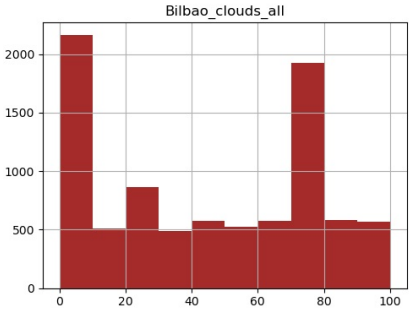
In [21]:
```
# Plot distribution of cateforical Variables
plot_categorical('Valencia_wind_deg', data= valencia)
```

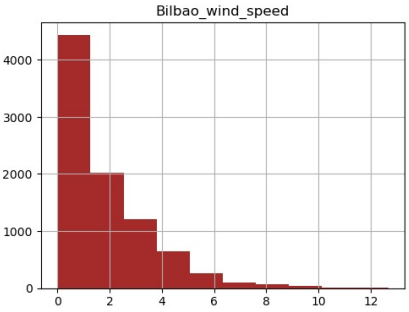


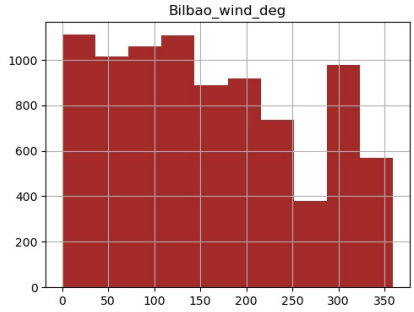
BILBAO

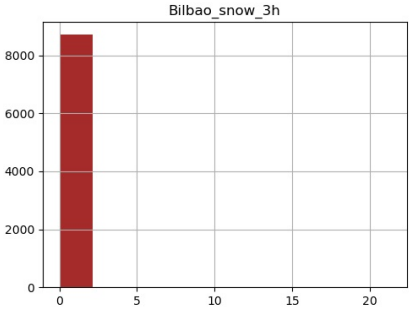
In [22]:
```
# # have a look at feature distributions Using histogram Plots
bilbao.hist(figsize=(20,20), color = 'brown');
```

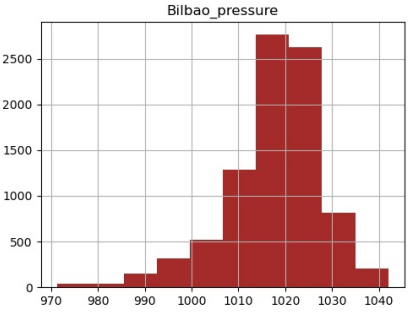


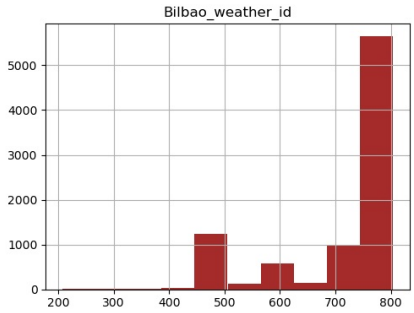


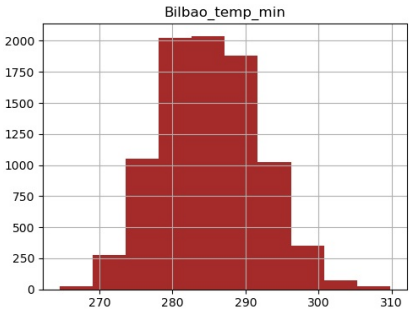


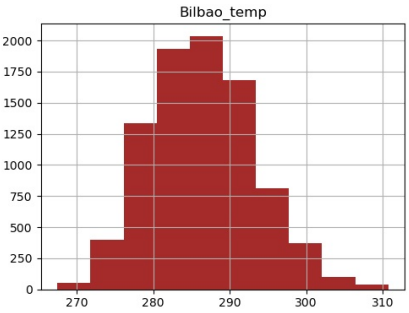


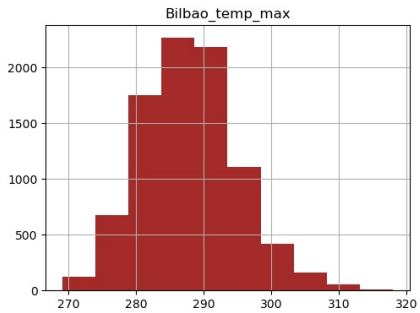


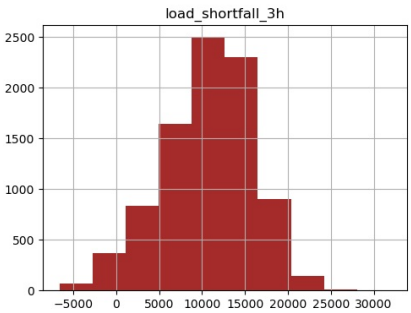




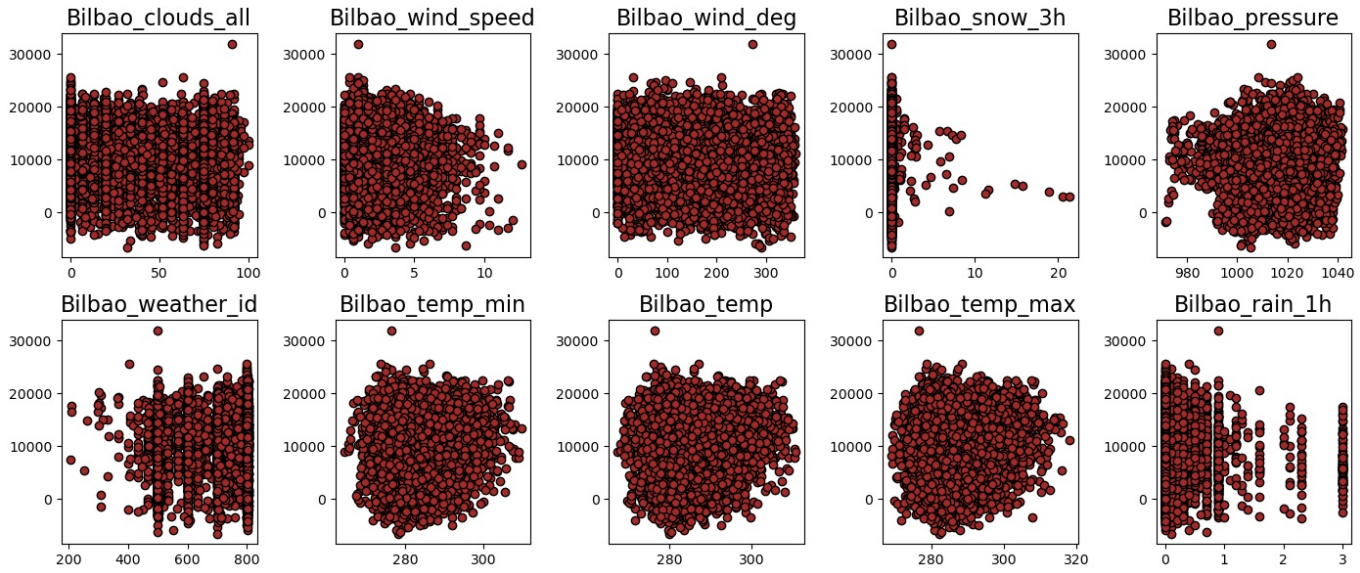




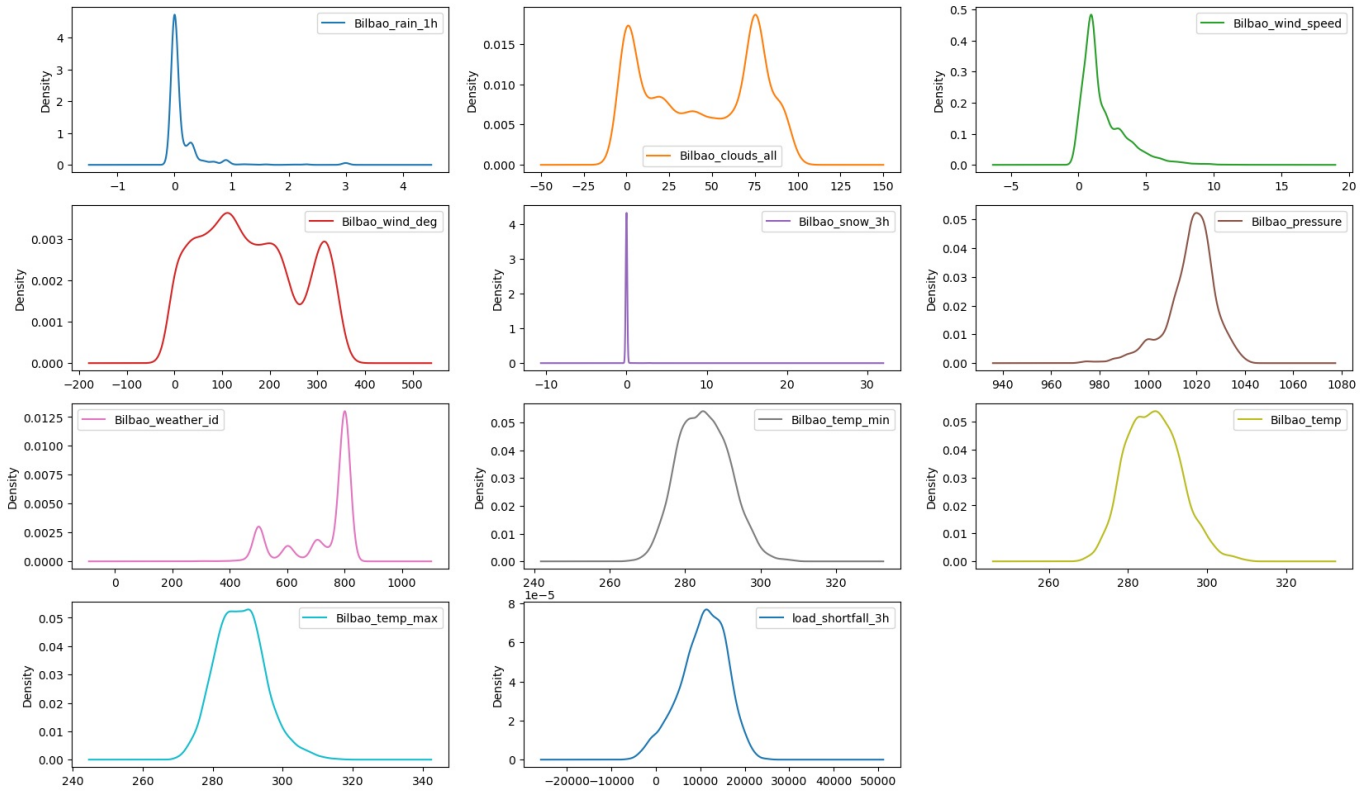




In [23]:
```
# Make Scatter Plots of Variables vs load_shortfall_3h
scatter_plot(bilbao, color = 'brown')
```



In [24]:
```
# have a look at feature distributions
bilbao.plot(kind= 'density', subplots=True, layout=(13,3), sharex= False, figsize= (20,40))
plt.show()
```

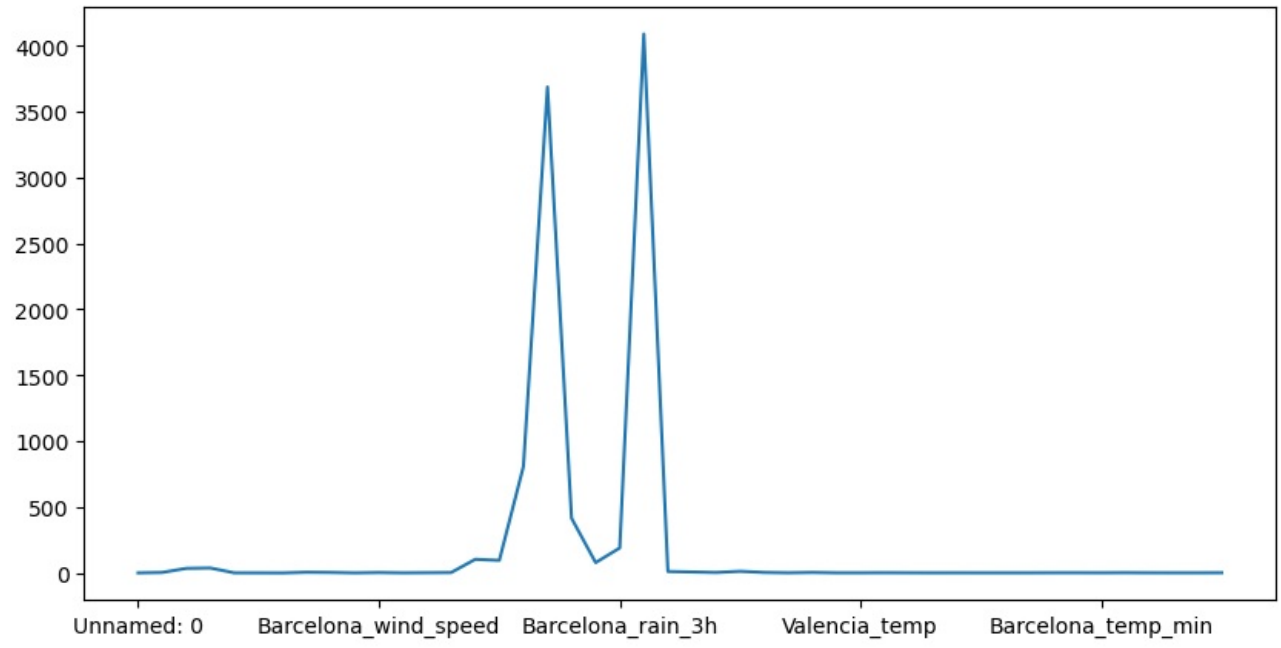


Checking for kurtosis

-Kurtosis measures how tailed our distribution is

In [25]:
```
#Check for kurtosis
plt.figure(figsize = [10,5])
train_data.kurtosis(numeric_only = True).plot()
```

Out[25]: <Axes: >



In [ ]:

In [26]:
```
#checking for outliers in each of the columns
train_data.kurtosis(numeric_only = True)
```

Out[26]:
```
Unnamed: 0                -1.200000
Madrid_wind_speed          2.036462
Bilbao_rain_1h            32.904656
Valencia_wind_speed       35.645426
Seville_humidity          -1.017983
Madrid_humidity           -1.167537
Bilbao_clouds_all         -1.533417
Bilbao_wind_speed          3.631565
Seville_clouds_all         2.155921
Bilbao_wind_deg           -1.083530
Barcelona_wind_speed       1.493635
Barcelona_wind_deg        -0.959160
Madrid_clouds_all          0.142079
Seville_wind_speed         1.398580
Barcelona_rain_1h        101.578931
Seville_rain_1h           93.840746
Bilbao_snow_3h           806.128471
Barcelona_pressure      3687.564230
Seville_rain_3h          413.136592
Madrid_rain_1h            76.584491
Barcelona_rain_3h        187.800460
Valencia_snow_3h        4089.323165
Madrid_weather_id          9.259047
Barcelona_weather_id       5.701882
Bilbao_pressure            1.825323
Seville_weather_id        10.710308
Valencia_pressure          2.211823
Seville_temp_max          -0.515989
Madrid_pressure            2.216199
Valencia_temp_max         -0.613755
Valencia_temp             -0.643793
Bilbao_weather_id          0.067814
Seville_temp              -0.504132
Valencia_humidity         -0.734345
Valencia_temp_min         -0.599551
Barcelona_temp_max        -0.728757
Madrid_temp_max           -0.662861
Barcelona_temp            -0.696555
Bilbao_temp_min           -0.230342
Bilbao_temp               -0.086363
Barcelona_temp_min        -0.474890
Bilbao_temp_max            0.283366
Seville_temp_min          -0.475564
Madrid_temp               -0.612299
Madrid_temp_min           -0.666646
load_shortfall_3h         -0.118999
dtype: float64
```

Kurtosis is a measure of the taildness of a distribution:

Below are features with large numbers of outliers as shown by kurtosis > 3:

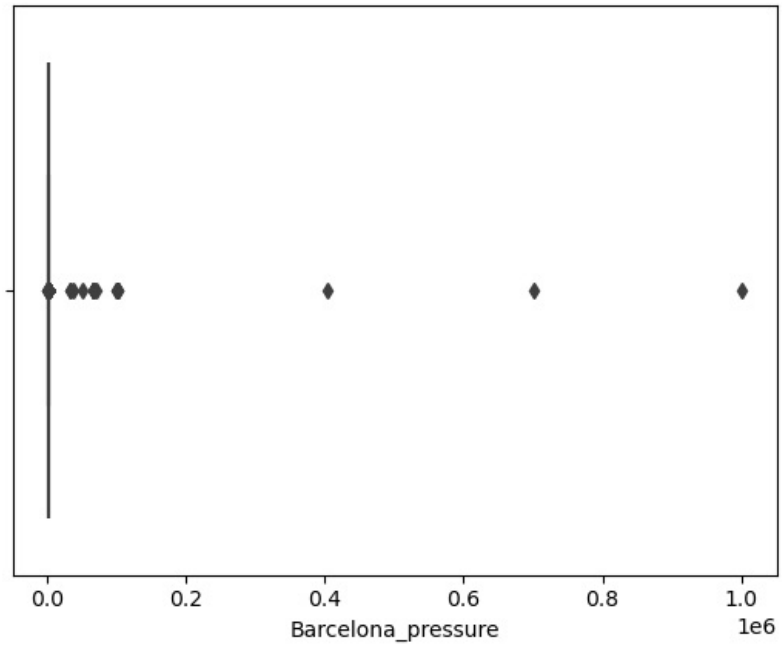
1.Bilbao_rain_1h, Valencia_wind_speed, Barcelona_rain_1h, Seville_rain_1h, Bilbao_snow_3h, Barcelona_pressure, Seville_rain_3h, Valencia_snow_3h, Barcelona_rain_3h, Madrid_weather_id, Barcelona_weather_id and Seville_weather_id.

2.The outliers that can be observed in Barcelona_pressure are due to some sort of error, as a pressure of 3687.564230 is too high and therfore unlikely to occur.We can therefore replace or drop the value in the data engineering section.

3.Changes in weather conditions on different days may cause outliers, which might explain the outliers in the other features hence we can leave them.

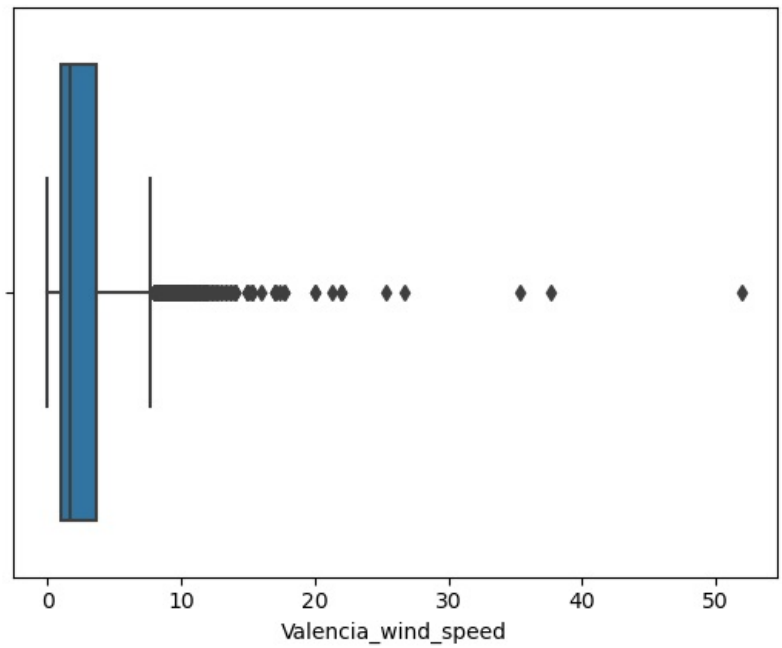
In [27]:
```
#Visual representation of the outliers
sns.boxplot(x = 'Barcelona_pressure', data = train_data)
```

Out[27]:
```
<Axes: xlabel='Barcelona_pressure'>
```



In [28]:
```
#Visual reprentation of the outliers present in the Valencia_wind_spead feature
sns.boxplot(x='Valencia_wind_speed', data=train_data)
```

Out[28]: <Axes: xlabel='Valencia_wind_speed'>



Now to check for skewness

-To measure the distortion of symmetrical distribution or asymmetry in our data set.

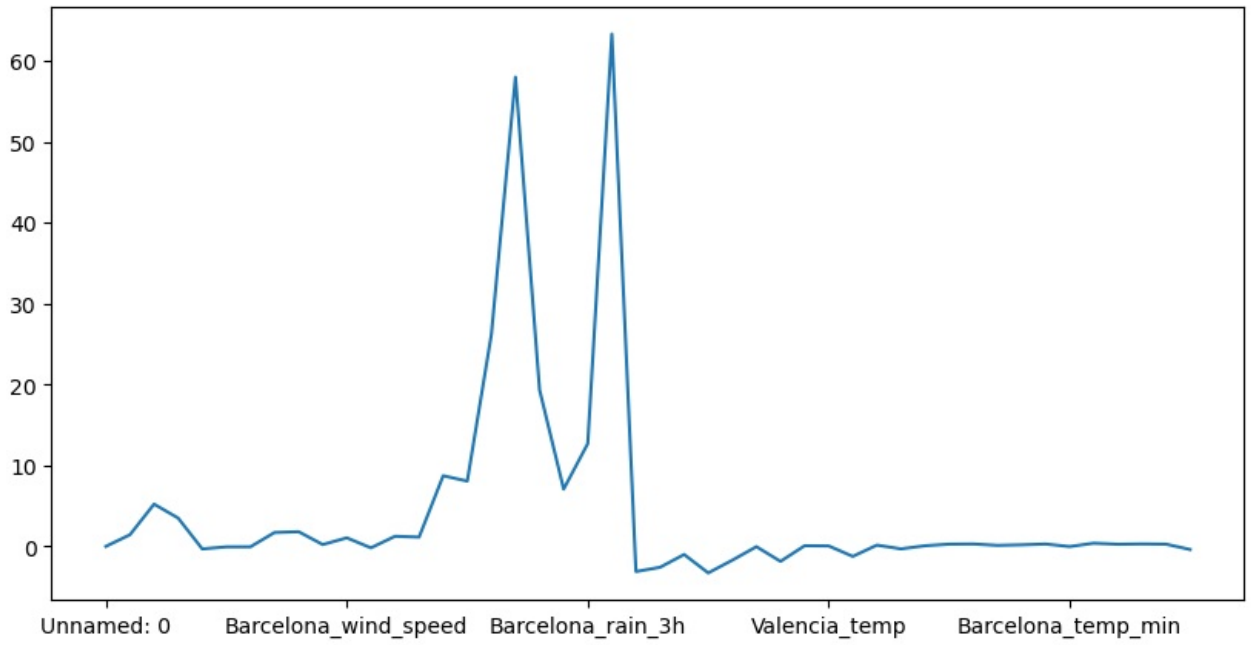
-if skewness < -1 or skewness > 1 then our data distribution is highly skewed

-if -1 < skewness < -0.5 or 0.5 < skewness < 1 then the distribution of our data is moderate

-if -0.5 < skewness < 0.5 then the distribution of our data is approximately symmetric

In [29]:
```
#Skewness of the data
plt.figure(figsize = [10,5])
train_data.skew(numeric_only = True).plot()
```

Out[29]: <Axes: >



In [30]:
```
#Checking how skewed our data is
train_data.skew(numeric_only = True)
```

Out[30]:
```
Unnamed: 0               0.000000
Madrid_wind_speed        1.441144
Bilbao_rain_1h           5.222802
Valencia_wind_speed      3.499637
Seville_humidity        -0.310175
Madrid_humidity         -0.057378
Bilbao_clouds_all       -0.053085
Bilbao_wind_speed        1.716914
Seville_clouds_all       1.814452
Bilbao_wind_deg          0.226927
Barcelona_wind_speed     1.057331
Barcelona_wind_deg      -0.180001
Madrid_clouds_all        1.246745
Seville_wind_speed       1.151006
Barcelona_rain_1h        8.726988
Seville_rain_1h          8.067341
Bilbao_snow_3h          26.177568
Barcelona_pressure      57.979664
Seville_rain_3h         19.342574
Madrid_rain_1h           7.074308
Barcelona_rain_3h       12.696605
Valencia_snow_3h        63.298084
Madrid_weather_id       -3.107722
Barcelona_weather_id    -2.584011
Bilbao_pressure         -0.999642
Seville_weather_id      -3.275574
Valencia_pressure       -1.705162
Seville_temp_max        -0.033931
Madrid_pressure         -1.850768
Valencia_temp_max        0.082672
Valencia_temp            0.057476
Bilbao_weather_id       -1.234844
Seville_temp             0.157238
Valencia_humidity       -0.305757
Valencia_temp_min        0.081832
Barcelona_temp_max       0.276925
Madrid_temp_max          0.298707
Barcelona_temp           0.128095
Bilbao_temp_min          0.194912
Bilbao_temp              0.293686
Barcelona_temp_min      -0.018057
Bilbao_temp_max          0.393932
Seville_temp_min         0.265482
Madrid_temp              0.304123
Madrid_temp_min          0.275083
load_shortfall_3h       -0.384847
dtype: float64
```

WHAT CAN WE OBSERVE FROM THE TABLE ABOVE?

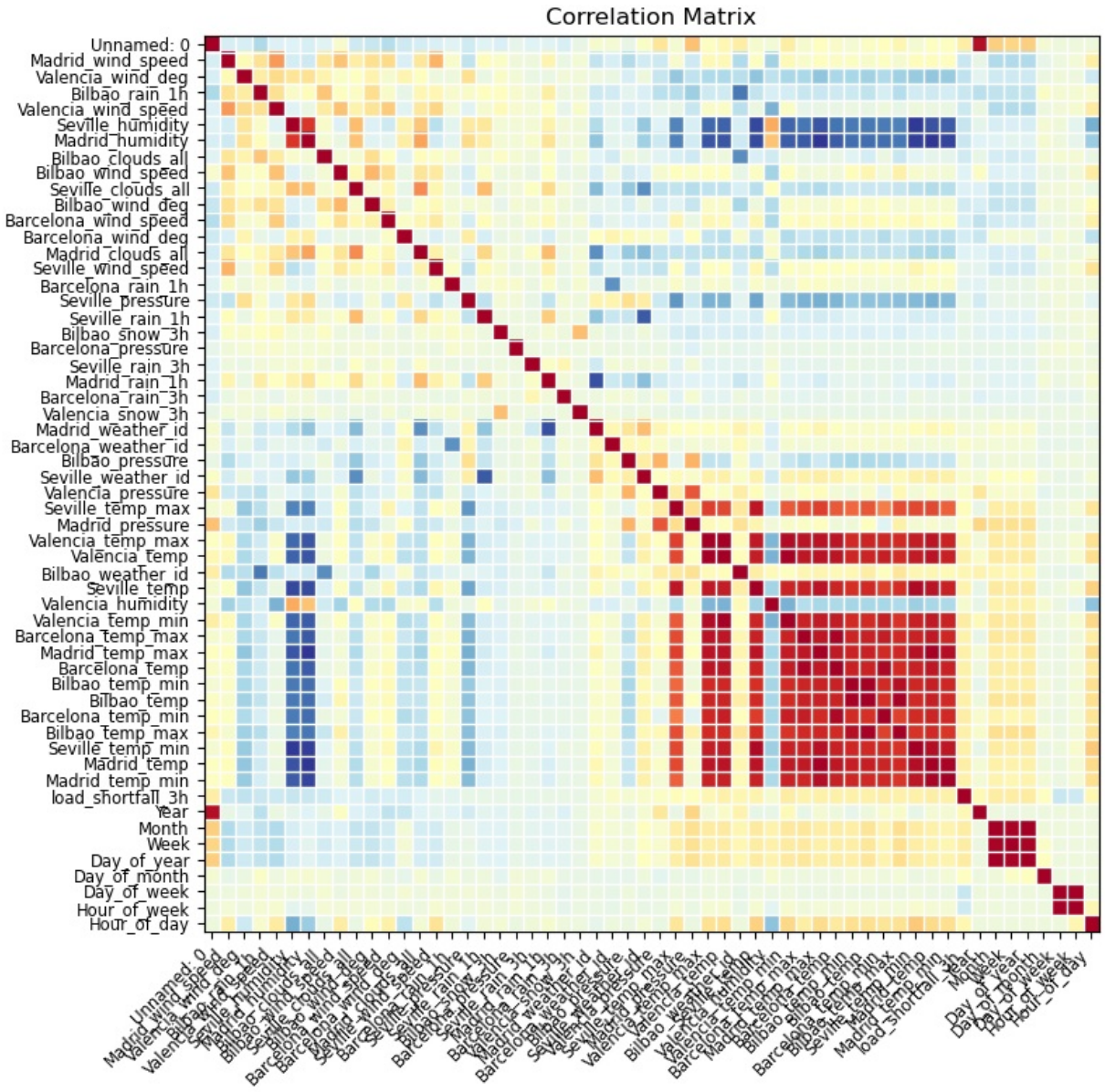
1.The following features have high positive symmetrical data; Bilbao_snow_3h, Barcelona_pressure, Seville_rain_3h, Barcelona_rain_3h and Valencia_snow_3h.

2.Madrid_weather_id, Barcelona_weather_id and Seville_weather_id have data with high negative symmetry. It is also noted that this features are also identified as outliers.

In [77]:
```
numeric_data_ = train_data.select_dtypes(include=[np.number])
```

In [79]:
```
#evaluate correlation

fig = plt.figure(figsize=(10,8));
ax = fig.add_subplot(111);
plot_corr(numeric_data_.corr(), xnames = numeric_data_.corr().columns, ax = ax, );
```



What can we observe from the visual representation of the correlation between features?

1.First, we can easily tell the presence of high correlation(in red) between features on the heatmap at the bottom right corner of our graph.

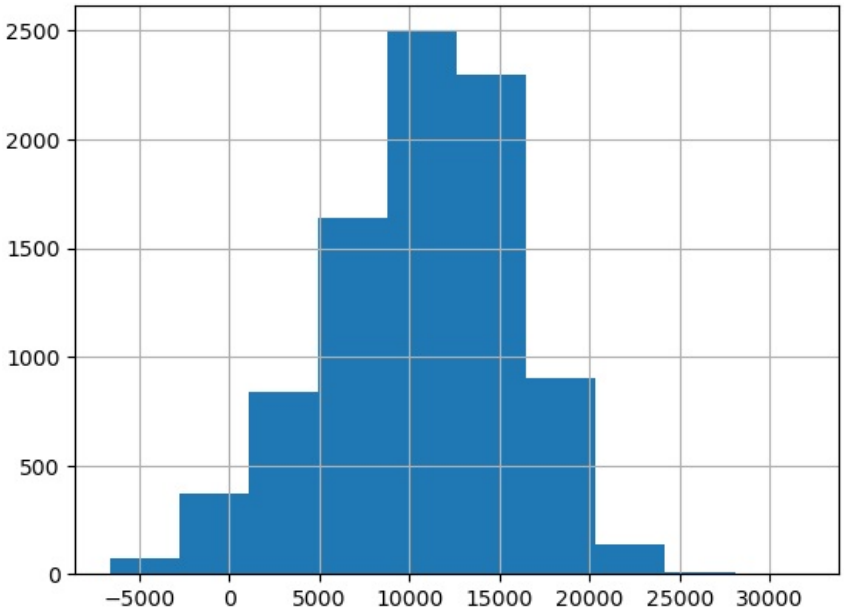
2.A breakdown of handling such occurence will be discussed in the feature engineering section of the notebook.

3.This step will be important when choosing the best features which in turn would play a hugerole in the improvement of our model.

In [32]:
```
# have a look at feature distributions

train_data.load_shortfall_3h.hist()
```

Out[32]:
```
<Axes: >
```



4. Data Engineering

Back to Table of Contents

⚡ **Description: Data engineering** ⚡

In this section you are required to: clean the dataset, and possibly create new features - as identified in the EDA phase.

KEY POINTS TO CONSIDER WHEN ENGINEERING OUR DATA:

1.Feature Selection/Importance(This is where we will choose relevent features and drop insignificant ones)

2.Handling missing values(Either we drop the entire column or fill in the the missing data)

3.Handling outliers

4.Feature Scaling(Where we will scale our dependent and independent variables)

In [33]:
```
#Making copy of train_data
clean_data = train_data
```

In [34]:
```
# remove missing values/ features
#Fill in the missing values of the 'Valencia_pressure' feature with the mean
clean_data['Valencia_pressure'].fillna(clean_data['Valencia_pressure'].mean(), inplace = True)
```

In [35]:
```
# Connvert Seville_pressure to numerical columns by scrapping off the 'sp' infrom of it

clean_data.Seville_pressure = clean_data.Seville_pressure.str[2:]
```

In [36]:
```
# Connvert Valencia_wind_deg to numerical columns by scrapping off the 'level_' infrom of it

```

```
clean_data.Valencia_wind_deg = clean_data.Valencia_wind_deg.str[6:]
```

In [37]:
```
#Type casting
clean_data['Valencia_wind_deg'] = clean_data['Valencia_wind_deg'].str.extract('(\d+)').astype('int64')
clean_data['Seville_pressure'] = clean_data['Seville_pressure'].str.extract('(\d+)').astype('int64')
```

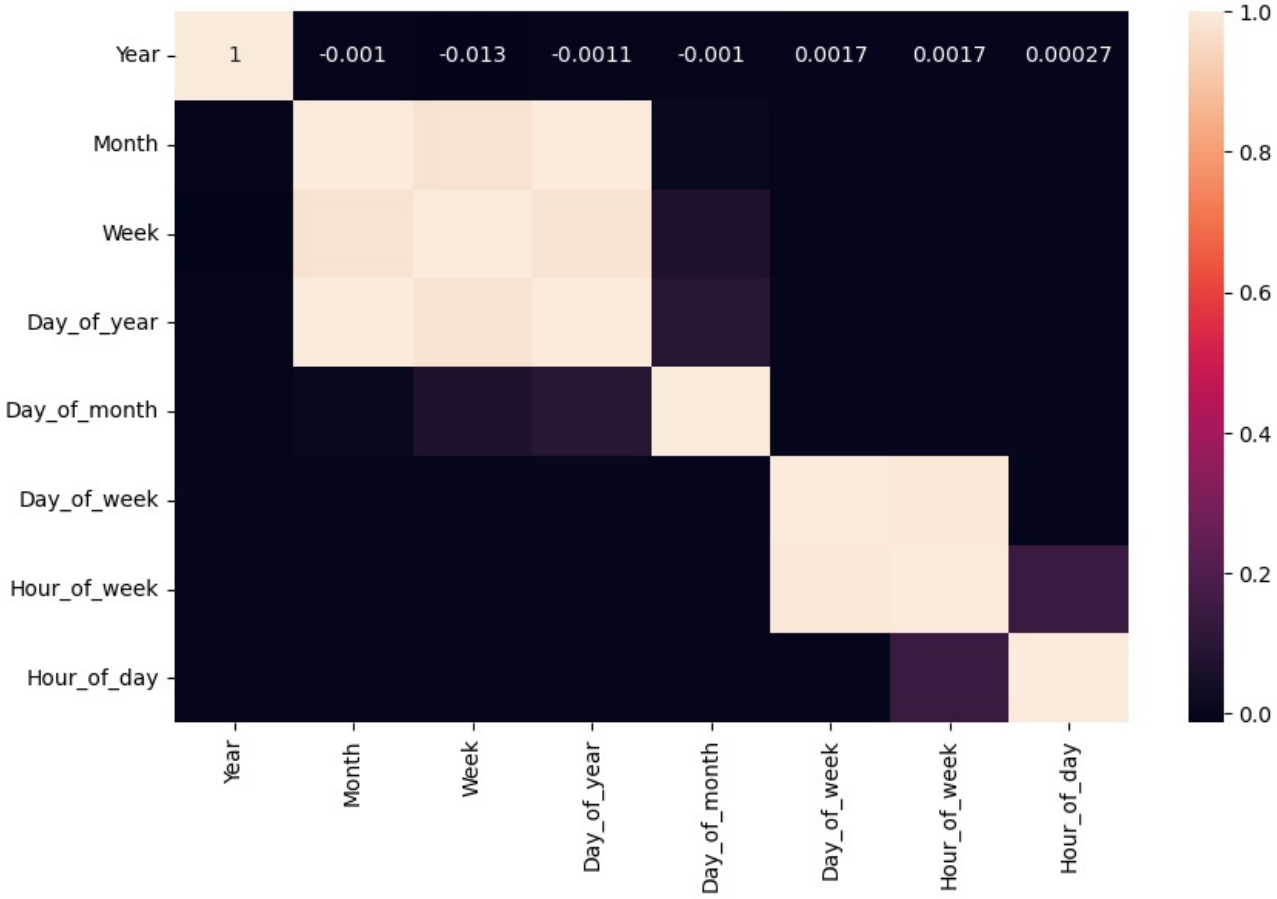
Below we will create and engineer our newly created features

In [40]:
```
# create new features
clean_data['Year']   = clean_data['time'].astype('datetime64[ns]').dt.year
clean_data['Month']  = clean_data['time'].astype('datetime64[ns]').dt.month
clean_data['Week'] = clean_data['time'].astype('datetime64[ns]').dt.isocalendar().week
clean_data['Day_of_year']  = clean_data['time'].astype('datetime64[ns]').dt.dayofyear
clean_data['Day_of_month']  = clean_data['time'].astype('datetime64[ns]').dt.day
clean_data['Day_of_week'] = clean_data['time'].astype('datetime64[ns]').dt.dayofweek
clean_data['Hour_of_week'] = ((clean_data['time'].astype('datetime64[ns]').dt.dayofweek) * 24 + 24) - (24 - tra
clean_data['Hour_of_day']  = clean_data['time'].astype('datetime64[ns]').dt.hour
```

We therefore visualise the correlation between our newly created features.

In [41]:
```
Time_df = clean_data.iloc[:,[-8,-7,-6,-5,-4,-3,-2,-1]]
#Time_df = clean_data.iloc[:,[-3,-2,-1]]
plt.figure(figsize=[10,6])
sns.heatmap(Time_df.corr(),annot=True )
```

Out[41]: <Axes: >



When looking at our heatmap we can clearly see that we have high Multicollinearity present in our new features.

The features with high Multicollinearity are:

1.Month_of_year

2.Week_of_year

3.Day_of_year

4.Day_of_week

5.Hour_of_week

Therefore either one of the feature which are highly correlated with another would have to be dropped.

We will then drop the time and Unnamed column alongside the features mentioned above.

In [42]:
```
#Drop some of the features
clean_data = clean_data.drop(columns=['Week','Day_of_year','Hour_of_week', 'Unnamed: 0','time'])
```

FEATURE SELECTION

Featue selection is a method used to reduce the input variable to our model by using only the relevant data and getting rid of noisy data.This is the process of choosing the relevant features for our model,this will be pivotal when building our model.

There are some methods for feature selection:

1.We can check the feature scores in our data,this score will help us in indicating which features ore relevent(The higher the score the more relevent that feature is)

2.Feature importance is a built-in class that comes with tree-based classifiers such as:

Random Forest Classifiers Extra Tree Classifiers

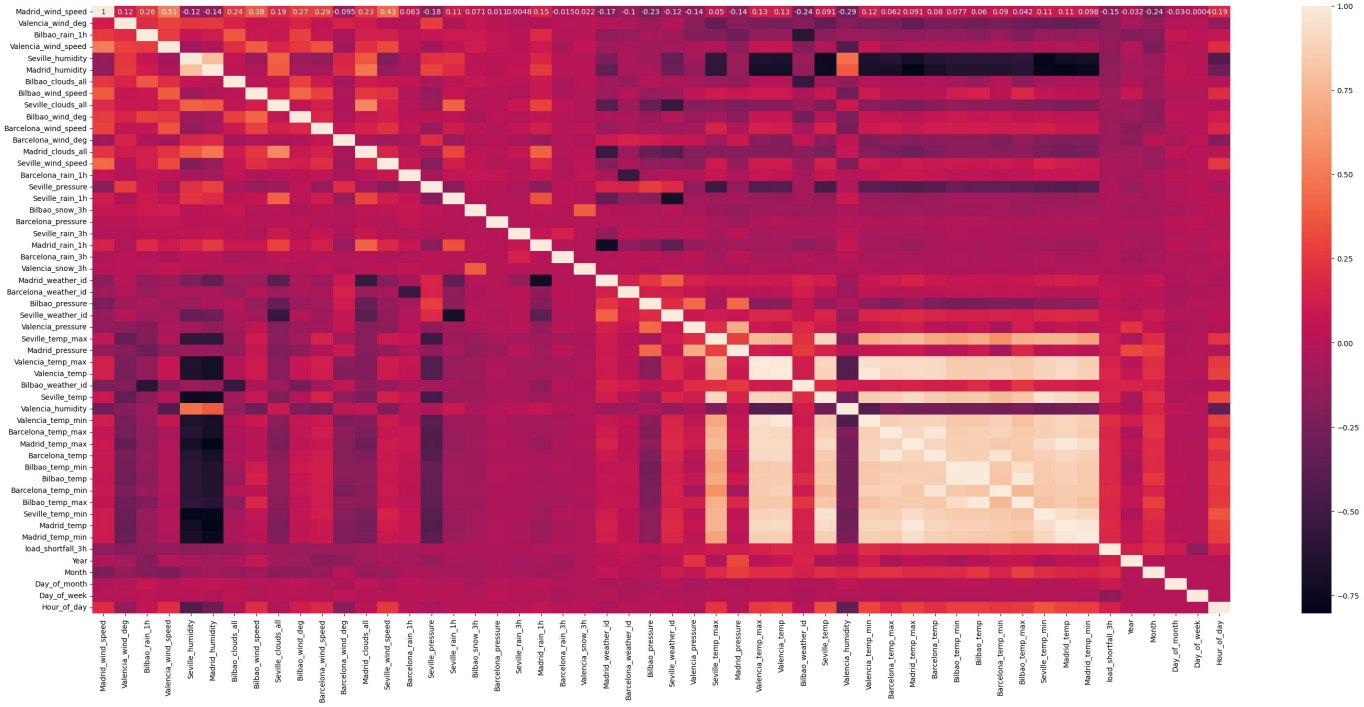
CORRELATION MATRIX WITH HEATMAP

Heatmap is a graphical representation of 2D (two-dimensional) data. Each data value represented in a matrix.

1.First, we'll plot the pair plot between all independent features and dependent features. It will give the relation between dependent and independent features. The relation between the independent feature and the dependent feature is less than 0.2 then choose that independent feature for building a model.

In [43]:
```
plt.figure(figsize=[35,15])
sns.heatmap(clean_data.corr(),annot=True )
```

Out[43]: <Axes: >



Here we notice the presence of high correlations between the independent variables(predictor variables) as well as possible outliers.

We will therefore have to drop the less significant columns for the benefit of improving our model perfomance and reducing the chances of overfitting our model. Before we drop these columns

In [44]:
```
#Get dummies
clean_data=pd.get_dummies(clean_data, drop_first=True)
```

WE WILL BE USING(SelectKBest and Chi2) TO PERFORM FEATURE SELECTION

In [45]:
```
#Split our data into independent and dependent variables
X = clean_data.drop(columns = 'load_shortfall_3h')
y = clean_data['load_shortfall_3h'].astype('int')
```

In [46]:
```
bestfeatures = SelectKBest(score_func=chi2, k=10)
fit = bestfeatures.fit(X,y)
dfscores = pd.DataFrame(fit.scores_)
dfcolumns = pd.DataFrame(X.columns)
featureScores = pd.concat([dfcolumns, dfscores], axis=1)
featureScores.columns = ['Features', 'Score']
new_X = featureScores.sort_values('Score',ascending=False)#.head(40)
new_X.tail(10) #To get the least important feature based on ther score
```

Out[46]:

	Features	Score
35	Valencia_temp_min	1148.530710
40	Bilbao_temp	1133.227759
39	Bilbao_temp_min	1101.915829
38	Barcelona_temp	1023.804735
41	Barcelona_temp_min	903.603041
25	Bilbao_pressure	682.609378
27	Valencia_pressure	473.028819
19	Seville_rain_3h	383.079442
21	Barcelona_rain_3h	247.853993
46	Year	2.314342

From the above table it is clear that the features with the least significance are the ones with the lowest score.This result also backups the result that we observed from our heatmap where we saw multicollinearity between certain features, Therefore we can see those features as having the lowest significance in our data.

SELECTING RELEVANT FEATURES FOR OUR MODEL:

1.Here we will select the only features that will help us in improving the perfomance of our model and by doing so we leave out the features which are unnecessary for building our model.

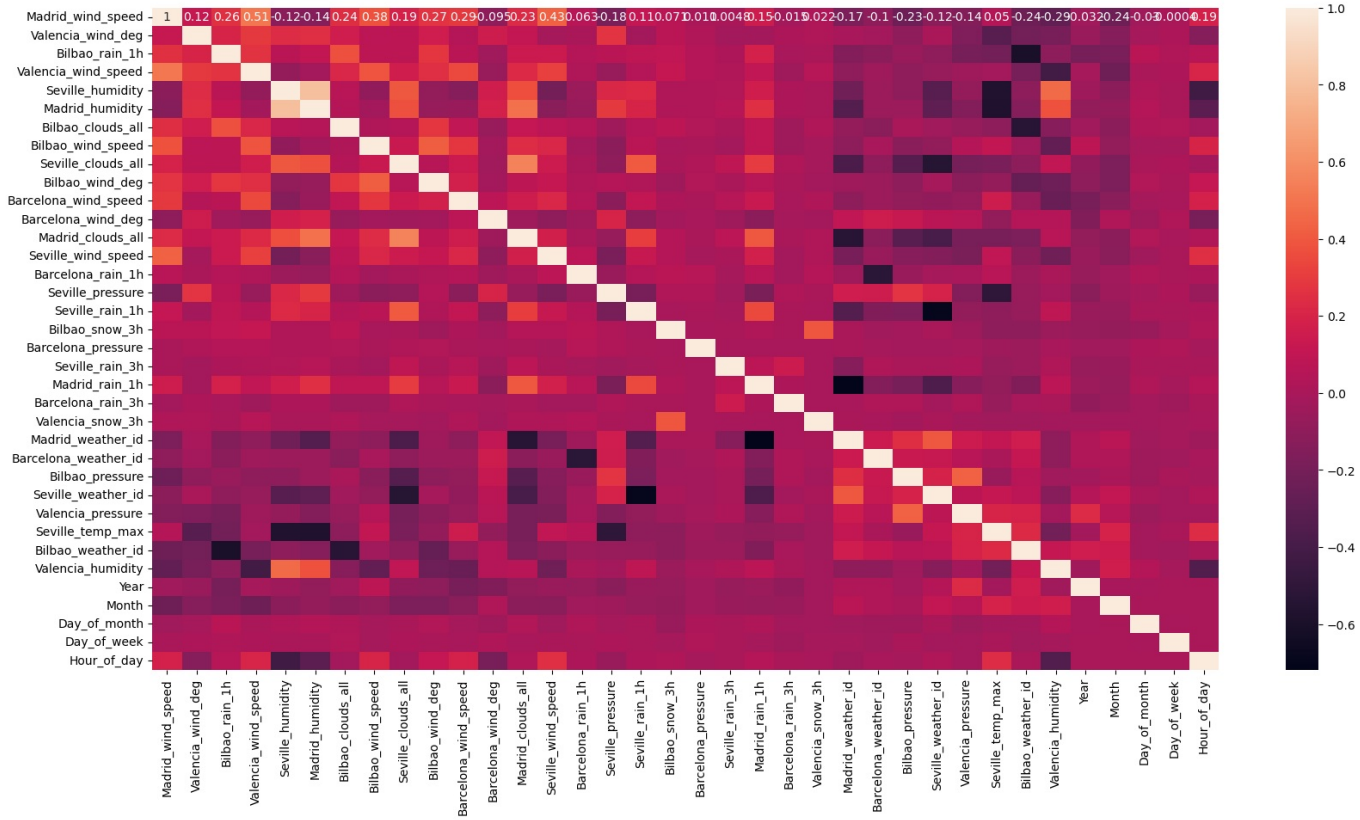
In [47]:
```
X = X[['Madrid_wind_speed', 'Valencia_wind_deg', 'Bilbao_rain_1h',
       'Valencia_wind_speed', 'Seville_humidity', 'Madrid_humidity',
       'Bilbao_clouds_all', 'Bilbao_wind_speed', 'Seville_clouds_all',
       'Bilbao_wind_deg', 'Barcelona_wind_speed', 'Barcelona_wind_deg',
       'Madrid_clouds_all', 'Seville_wind_speed', 'Barcelona_rain_1h',
       'Seville_pressure', 'Seville_rain_1h', 'Bilbao_snow_3h',
       'Barcelona_pressure', 'Seville_rain_3h', 'Madrid_rain_1h',
       'Barcelona_rain_3h', 'Valencia_snow_3h', 'Madrid_weather_id',
       'Barcelona_weather_id', 'Bilbao_pressure', 'Seville_weather_id',
       'Valencia_pressure', 'Seville_temp_max', 'Bilbao_weather_id',
       'Valencia_humidity', 'Year', 'Month', 'Day_of_month', 'Day_of_week', 'Hour_of_day']]
```

In [ ]:
```
```

In [48]:
```
#Heatmap of the selected features

plt.figure(figsize=[20,10])
sns.heatmap(X.corr(),annot=True )
```

Out[48]: <Axes: >



What can we observe from the above heatmap ?

It is clear that by selecting the relevant features we were able to reduce the multicollinearity in our data.

NOW WE CAN PERFORM FEATURE SCALING:

What is feature scaling? Feature scaling is a method used for mormalising the range of independent variables or the features of the data.Real-world datasets usually contain features of varying degrees of magnitude,range and units.Therefore we need to perform feature scaling in order to enable our model to interpret these features on the scale.

Therefore we will be using standard scaler as it is robust to outliers.

In [49]:
```
#Create standart scaler
scaler = StandardScaler()
```

In [50]:
```
#Store the standardised features in a new variable X_scaled

X_scaled = scaler.fit_transform(X)
X_scaled = pd.DataFrame(X_scaled,columns=X.columns)
X_scaled.head()
```

Out[50]:

	Madrid_wind_speed	Valencia_wind_deg	Bilbao_rain_1h	Valencia_wind_speed	Seville_humidity	Madrid_humidity	Bilbao_clouds_
0	-0.950708	-0.096053	-0.362123	-0.796169	0.516117	0.270621	-1.3354
1	-1.130863	1.641580	-0.362123	-0.381412	0.692953	0.298017	-1.3354
2	-0.770554	1.294054	-0.362123	-0.657917	0.383491	0.284319	-1.3354
3	-0.770554	0.946527	-0.362123	-0.657917	0.118238	-0.044439	-1.3354
4	-0.770554	0.599000	-0.362123	-0.657917	-0.161751	-0.017043	-1.2740

5 rows × 36 columns

In [ ]:

5. Modelling

Back to Table of Contents

⚡ **Description: Modelling** ⚡

In this section, you are required to create one or more regression models that are able to accurately predict the thee hour load shortfall.

MODEL BUILDING

1.We will first split our data(using train-test split)

2.fit onto train data

3.predict on test model

4.evaluate performance

In [51]:
```
# split data
X_train, X_test, y_train, y_test = train_test_split(X_scaled, y, test_size = 0.33, random_state = PARAMETER_CONS
```

In [52]:
```
#checking the shape of the training and testing data

print('Training predictor:', X_train.shape)
print('Training target:', y_train.shape)
print('Testing predictor:', X_test.shape)
print('Testing target:', y_test.shape)
```

```
Training predictor: (5871, 36)
Training target: (5871,)
Testing predictor: (2892, 36)
Testing target: (2892,)
```

In [53]:
```
#Create one or more ML models
#Instantiate the model

lm = LinearRegression() #regression model
RF = RandomForestRegressor(n_estimators = 150, random_state = PARAMETER_CONSTANT) #Random forest model
Reg_tree = DecisionTreeRegressor(random_state = PARAMETER_CONSTANT ) #Decision Tree
```

In [54]:
```
# Fit into taining data

lm.fit(X_train, y_train)
Reg_tree.fit(X_train,y_train)
RF.fit(X_train,y_train)
```

Out[54]:
```
RandomForestRegressor(n_estimators=150, random_state=42)
```

In [55]:
```
#Predicting on test model

lm_pred = lm.predict(X_test)
RF_pred = RF.predict(X_test)
Tree_pred = Reg_tree.predict(X_test)

#Predictng on the same training set
train_predict = lm.predict(X_train)
```

Now we can compare the trues values and the values predicted by our models

In [56]:
```
# evaluate one or more ML models
#Comparing the True value and the Predicted Value of our models
Linear_model = pd.DataFrame({'Actual': y_test, 'Predicted': lm_pred})
Tree_model = pd.DataFrame({'Actual': y_test, 'Predicted': Tree_pred})
Forest_model = pd.DataFrame({'Actual': y_test, 'Predicted': RF_pred})
```

In [57]:
```
#Linear model actual and predicted Value
Linear_model.head()
```

Out[57]:

	Actual	Predicted
1226	11450	12419.462193
7903	13693	12270.999632
1559	18337	12080.498644
3621	-1221	9286.623093
7552	8515	14425.974867

In [58]:
```
#Regression tree model actual and predicted value
Tree_model.head()
```

Out[58]:

	Actual	Predicted
1226	11450	3232.0
7903	13693	12355.0
1559	18337	17458.0
3621	-1221	574.0
7552	8515	12596.0

In [59]:
```
#Random forest model actual and predicted value
Forest_model.head()
```

Out[59]:

	Actual	Predicted
1226	11450	7655.480000
7903	13693	13419.646667
1559	18337	16296.306667
3621	-1221	2477.426667
7552	8515	10937.433333

6. Model Performance

Back to Table of Contents

⚡ **Description: Model performance** ⚡

In this section you are required to compare the relative performance of the various trained ML models on a holdout dataset and comment on what model is the best and why.

What to do now?

Now we can evaluate the performance of the models we created and then choose the model which performs a lot better.

How are we going to do that?

We can use the metrics below to check model performane:

1.Root Mean Squared Error (RMSE)

2,Mean Squared Error (MSE)

3.Mean Absolute Error (MAE)

4.R Squared (R^2)

In [60]:
```
#Create dictionary for test RMSE
Performance_RMSE = {

                        'Test RMSE':

                          {"Linear model": np.sqrt(metrics.mean_squared_error(y_test,lm_pred)),
                          "Decision Tree" : np.sqrt(metrics.mean_squared_error(y_test,Tree_pred)),
                          "Random Forest" : np.sqrt(metrics.mean_squared_error(y_test,RF_pred))}

                      }

# create dataframe of the dictionary
Performance_RMSE = pd.DataFrame(data=Performance_RMSE)
Performance_RMSE
```

Out[60]:

	Test RMSE
Decision Tree	3990.150135
Linear model	4826.678006
Random Forest	2883.447339

In [61]:
```
#Create visuals of results
px.bar(Performance_RMSE, y =Performance_RMSE['Test RMSE'],
       color = Performance_RMSE.index, width =700, height=400)
```

What can be observed from the above findings?

We can see that the Random Forest model outperforms the othe models with a lower RMSE value.

In [62]:
```
#Create dictionary for the test MSE
Performance_MSE = {

                        'Test MSE':

                          {"Linear model": (metrics.mean_squared_error(y_test,lm_pred)),
                          "Decision Tree" : (metrics.mean_squared_error(y_test,Tree_pred)),
                          "Random Forest" : (metrics.mean_squared_error(y_test,RF_pred))}

                      }

# create dataframe of dictionary
Performance_MSE = pd.DataFrame(data=Performance_MSE)
Performance_MSE
```

Out[62]:

	Test MSE
Decision Tree	1.592130e+07
Linear model	2.329682e+07
Random Forest	8.314269e+06

In [63]:
```
#Create visuals of the results
px.bar(Performance_MSE, y =Performance_MSE['Test MSE'],
       color = Performance_MSE.index, width =700, height=400)
```

What can we observe from the above findings?

We are able to see the values of the test MSE as well as their visual representations, we can therefore see that the Random Forest model perfoms better with a lower MSE value as compared to the othe models.

In [64]:
```
#Create dictionary of the Test Mean Absolute Error(MAE)
Performance_MAE= {

                      'Test MAE':

                        {"Linear model": (metrics.mean_absolute_error(y_test,lm_pred)),
                        "Decision Tree" : (metrics.mean_absolute_error(y_test,Tree_pred)),
                        "Random Forest" : (metrics.mean_absolute_error(y_test,RF_pred))}

                    }

# create dataframe of dictionary
Performance_MAE = pd.DataFrame(data = Performance_MAE)
Performance_MAE
```

Out[64]:

	Test MAE
Decision Tree	2931.309474
Linear model	3828.307237
Random Forest	2208.181215

In [65]:
```
#Create visuals of results
px.bar(Performance_MAE, y = Performance_MAE['Test MAE'],
       color = Performance_MAE.index, width =700, height=400)
```

What can be observed from the above findings?

We can see that the Random Forest model performs better based on the lower MAE value

In [66]:
```
#Create dictionary for test R^2
Performance_Rsquared= {

                       'Test R^2':

                         {"Linear model": (metrics.r2_score(y_test,lm_pred)),
                          "Decision Tree" : (metrics.r2_score(y_test,Tree_pred)),
                          "Random Forest" : (metrics.r2_score(y_test,RF_pred))}

                     }

# create dataframe of dictionary

Performance_Rsquared = pd.DataFrame(data = Performance_Rsquared)
Performance_Rsquared
```

Out[66]:

	Test R^2
Decision Tree	0.429527
Linear model	0.165256
Random Forest	0.702093

In [67]:
```
#Create visuals of results
px.bar(Performance_Rsquared, y = Performance_Rsquared['Test R^2'],
       color = Performance_Rsquared.index, width =700, height=400)
```

What can we observe from the above findings?

We can see that the Random Forest model has a higher R^2 valuue as compared to the other models,which indicates that the Random Forest model performs better than the others.

NOW WHAT IS THE NEXT STEP?

Choose best model and motivate why it is the best choice:

Therefore the chosen model will have to be the Random Forest model and the reasons would be:

1.It performs better in terms of the low RMSE and MSE values which indicates that the avarage difference between the actual values and the values predicted by the model is less as compared to the other models, hence it performs best.

2.It also has a low Mean Absolute Error(MAE) value,which indicates a more accurate prediction .

3.The R square of the Random Forest model is higher than of the other models, which indicates how well the model will fit our data.

With all that being said I can conclude that the best model for our predictions will have to be the Random Forest model.

PREPAIRING OUR CHOSEN MODEL FOR KAGGLE SUBMISSION

In [68]:
```
test_data.head()
```

Out[68]:

	Unnamed: 0	time	Madrid_wind_speed	Valencia_wind_deg	Bilbao_rain_1h	Valencia_wind_speed	Seville_humidity	Madrid_hur
0	8763	2018-01-01 00:00:00	5.000000	level_8	0.0	5.000000	87.000000	71.3
1	8764	2018-01-01 03:00:00	4.666667	level_8	0.0	5.333333	89.000000	78.0
2	8765	2018-01-01 06:00:00	2.333333	level_7	0.0	5.000000	89.000000	89.6
3	8766	2018-01-01 09:00:00	2.666667	level_7	0.0	5.333333	93.333333	82.6
4	8767	2018-01-01 12:00:00	4.000000	level_7	0.0	8.666667	65.333333	64.0

5 rows × 48 columns

REPEAT THE PREPROCESSING AND CLEANING ON THE TEST DATA

In [69]:
```
output = pd.DataFrame({"time":test_data['time']})

#filling in missing values
test_data['Valencia_pressure'].fillna(test_data['Valencia_pressure'].mean(), inplace = True)

# Convert Seville_pressure to numerical columns by scrapping off the 'sp' infrom of it and also convert Valencia
test_data.Seville_pressure = test_data.Seville_pressure.str[2:]
test_data.Valencia_wind_deg = test_data.Valencia_wind_deg.str[6:]


#Typecasting
test_data['Valencia_wind_deg'] = test_data['Valencia_wind_deg'].str.extract('(\d+)').astype('int64')
test_data['Seville_pressure'] = test_data['Seville_pressure'].str.extract('(\d+)').astype('int64')
```

In [71]:
```
#Engineering New Features

test_data['Year']  = test_data['time'].astype('datetime64[ns]').dt.year
test_data['Month']  = test_data['time'].astype('datetime64[ns]').dt.month
test_data['Week'] = test_data['time'].astype('datetime64[ns]').dt.isocalendar().week
test_data['Day_of_year']  = test_data['time'].astype('datetime64[ns]').dt.dayofyear
```

```
test_data['Day_of_month']  = test_data['time'].astype('datetime64[ns]').dt.day
test_data['Day_of_week'] = test_data['time'].astype('datetime64[ns]').dt.dayofweek
test_data['Hour_of_week'] = ((test_data['time'].astype('datetime64[ns]').dt.dayofweek) * 24 + 24) - (24 - test_
test_data['Hour_of_day']  = test_data['time'].astype('datetime64[ns]').dt.hour
```

In [72]:
```
#Drop the 'Unnamed' and the 'time' features
test_data = test_data.drop(columns=['Week','Day_of_year','Hour_of_week', 'Unnamed: 0','time'])
```

In [73]:
```
#Select significant features
test_data = test_data[['Madrid_wind_speed', 'Valencia_wind_deg', 'Bilbao_rain_1h',
       'Valencia_wind_speed', 'Seville_humidity', 'Madrid_humidity',
       'Bilbao_clouds_all', 'Bilbao_wind_speed', 'Seville_clouds_all',
       'Bilbao_wind_deg', 'Barcelona_wind_speed', 'Barcelona_wind_deg',
       'Madrid_clouds_all', 'Seville_wind_speed', 'Barcelona_rain_1h',
       'Seville_pressure', 'Seville_rain_1h', 'Bilbao_snow_3h',
       'Barcelona_pressure', 'Seville_rain_3h', 'Madrid_rain_1h',
       'Barcelona_rain_3h', 'Valencia_snow_3h', 'Madrid_weather_id',
       'Barcelona_weather_id', 'Bilbao_pressure', 'Seville_weather_id',
       'Valencia_pressure', 'Seville_temp_max', 'Bilbao_weather_id',
       'Valencia_humidity', 'Year', 'Month', 'Day_of_month', 'Day_of_week', 'Hour_of_day']]
```

In [ ]:
```
```

In [74]:
```
test_data = scaler.fit_transform(test_data)
test_data = pd.DataFrame(test_data,columns=X.columns)
test_data.head()
```

Out[74]:

	Madrid_wind_speed	Valencia_wind_deg	Bilbao_rain_1h	Valencia_wind_speed	Seville_humidity	Madrid_humidity	Bilbao_clouds_
0	1.432590	0.703345	-0.440268	0.995600	0.964514	0.360022	-0.7662
1	1.244747	0.703345	-0.440268	1.162600	1.061565	0.636255	-1.4223
2	-0.070152	0.337515	-0.440268	0.995600	1.061565	1.119662	-1.4223
3	0.117691	0.337515	-0.440268	1.162600	1.271842	0.829617	-0.5475
4	0.869062	0.337515	-0.440268	2.832609	-0.086869	0.056166	-0.5475

5 rows × 36 columns

In [75]:
```
output['load_shortfall_3h'] = RF.predict(test_data)
output.head()
```

Out[75]:

	time	load_shortfall_3h
0	2018-01-01 00:00:00	7441.520000
1	2018-01-01 03:00:00	5018.873333
2	2018-01-01 06:00:00	4847.180000
3	2018-01-01 09:00:00	6579.060000
4	2018-01-01 12:00:00	7158.066667

CONVERSION TO CSV FILE

In [76]:
```
#convert output to csv
output.to_csv("submission_sample2.csv",index=False)
pd.read_csv("submission_sample2.csv").head()
```

Out[76]:

	time	load_shortfall_3h
0	2018-01-01 00:00:00	7441.520000
1	2018-01-01 03:00:00	5018.873333
2	2018-01-01 06:00:00	4847.180000
3	2018-01-01 09:00:00	6579.060000
4	2018-01-01 12:00:00	7158.066667

7. Model Explanations

Back to Table of Contents

⚡ **Description: Model explanation** ⚡

In this section, you are required to discuss how the best performing model works in a simple way so that both technical and non-technical

In this section, you are required to discuss how the best performing model works in a simple way so that both technical and non-technical stakeholders can grasp the intuition behind the model's inner workings.

Explanation of Performance Indicators

1.Coefficient of determination (R^2) mwhich determines thee proportion of variance in the dependent variable that can be explained by the independent variable.

-Rsquared values range from 0 to 1, where a high R squared (R^2 > 0.5) signifies a good fit and R^2 = 1 indicates a perfect fit.

2.RMSE is the square root of the mean square error (MSE) which is a measure that evaluates the quality of predictions, it measures the avarage difference between the actual values and the values predicted by the model.

-MSE measures the variance of the residuals, while the RMSE measures the standard deviation of the residuals. The lower the RMSE the better our model will perform.

discuss chosen methods logic

Random forests addresses the problem of overfitting, by growing multiple trees which are merged together for a more accurate prediction, the multiple uncorrelated models(individual decision trees) perform much better when in a group than they do alone.

They use ensemble learning methods for regression by constructing several Decision trees during training and outputs the average mean of the classes as the prediction of all the trees.

From our model comparisions, Random Forest gives us the highest R^2 and the lowest RMSE, which ,makes it the best performing model when compared to the others.

Conclusion

As an aspiring data scientist I have managed to developed a model that will assist in predicting the 3 hourly load shortfall between the energy generated by means of fossil fuels and various renewable sources.

This will assist in informing the Spanish Government of any trends and patterns of the country's renewable resources and fossil fuel energy generation and whether it is viable to expand its' renewable energy resource infrastructure investments

In [ ]: